TRINITY COLLEGE DUBLIN
COLÁISTE NA TRÍONÓIDE
THE UNIVERSITY OF DUBLIN

UML based Semantic Mapping Creation Tool

by

Neeraj Dixit

Dissertation

presented to the

Trinity College, University of Dublin

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

September, 2014

Supervisor

Dr. Declan O'Sullivan

Knowledge and Data Engineering Group

School of Computer Science & Statistics, Trinity College Dublin

Declaration

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

_____

Neeraj Dixit
August 27, 2014

Permission to lend and/or copy

I agree that Trinity College Library may lend or copy this dissertation upon request subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgment.

_____

Neeraj Dixit
August 27, 2014

Acknowledgements

First and foremost I would like to thank my supervisor Dr. Declan O'Sullivan for agreeing to supervise my research, for his guidance, support and insightful contributions to this research. I would also like to acknowledge all the volunteers who have given their time to participate in the experiments of this research.

Abstract

Knowledge models can be used to share domain knowledge between systems and users, and adding Semantics to data is seen as the solution provider to enable interoperability and integration of diverse software systems. Ontologies are the key factors to adding such semantics, but ontologies produced by different systems and different people show high heterogeneity which hinders interoperability. Semantic Mapping could provide a way by which varied ontologies could be used to exchange this semantic information. However manual mapping generation and development is difficult, tedious, error prone and requires knowledge of ontology languages and mapping techniques and so knowledge engineers often perform these tasks. However, domain experts with good domain knowledge are better suited for such tasks. But, most domain experts and system engineers find it too difficult or lack the knowledge of mapping tools and techniques required to perform semantic mappings. This dissertation presents a UML based tool to create semantic mapping. UML class diagrams which are commonly used by domain experts and system engineers for modeling domains and so the tool tries to provide an interface which represents mappings in a similar manner which is easy to understand and use by domain experts and also generates mappings that can be used by them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Over the years, ontologies have gained importance in enterprise software systems as they become more connected and integrate data from varied sources. The ontologies however, could be developed differently, making them incompatible with each other. This problem of incompatibility between ontologies representing same or similar domain calls for creating mappings between the involved ontologies. Since creating mappings is a difficult, time consuming, and require understanding of the knowledge engineering process, lots of research (Choi et al [Choi2006], Madhavan et al [Madhavan2002], Stabb et al [Staab2004]) has been undertaken that deal with approaches of mapping generation while others Ngo et al [Ngo2013] present the matching techniques used to derive correspondence between entities.

Since semantic mapping is a specialized field, knowledge engineers often work on creating semantic mappings. But as the enterprise systems become more complex the knowledge engineers often fall short of the domain knowledge required to sufficiently map the domains. This problem exposes the need for mappings to be done by domain experts and/or system/software engineers. Though these domain experts and system engineers have sufficient domain knowledge they do not have enough knowledge in ontology and semantic mapping techniques and so neither the knowledge engineers nor the system engineers are able to sufficiently complete the mapping process. In order to solve this problem efforts have been made to allow system engineers to use the tools well understood by them to create and represent mappings that can be used for integration. A number of works (Baclawski et al [Baclawski2001], Wang et al [Wang2001], Fu et al [Fu2009]), have tried

to use Unified Modeling Language (UML) for ontology modeling and to extract knowledge from them. Most of the existing approaches try to use UML for ontology modeling, which still need the system engineers to have good knowledge of ontology mapping techniques. Hence this research will focus on developing a system that automatically creates mappings from UML class diagrams and representing them in a way that is easy to use, understand and manipulate by a typical system engineer. Also, since completely automatic mapping generation with ability to generate all the mappings is considered infeasible (Noy et al [Noy2004]), the requirement for user input is important and stressed in mapping generation process Falconer et al ([Falconer2009], Shvaiko et al [Shvaiko2008]). Hence this research focuses on first providing ways with which users can provide inputs for creating mappings and then developing a natural and easy to use interface to allow system engineers to not only edit generated mappings but also create more mappings manually with the interface as part of the overall process.

1.2 Research Question

The research question addressed by the dissertation is:

To what extent can a semantic mapping representation based on UML be usable by software/system engineers?

Semantic mapping is defined as a process of finding correlations between different semantic models Kalfoglou et al [Kalfoglou2003] and mappings can be represented using different ontology techniques. Considering that most system/software engineers have little background in such techniques this research focuses on creating a tool that can help system engineers to easily generate mappings using unified modeling language (UML), an existing modeling technique used extensively by them. To create an interface that makes is easier and more natural for system engineers to create and manipulate mappings. In this research the mapping types considered for the tool are restricted to simple direct mappings and complex direct mappings.

1.3 Research Objectives and Goals

Following objectives are derived from the research objective.

1. Survey of the state of the art on various aspects involved in the research

The objective is to review existing literature to determine up to date research and work in the field of ontology matching and mapping techniques, ontology modeling with UML, ontology modeling with OWL and generating OWL from UML. The goal is to identify the limitations of current research and also investigate their usefulness to various components required for this research.

2. Develop an approach to obtain knowledge models from UML and develop mapping generation

   This objective requires developing an approach that is more suitable for system/software engineers than knowledge engineers. To achieve this goal, the approach of using UML class diagrams as starting point along with the rules provided by the users to obtain model knowledge is used and the rest of the approach is built on top of it, first converting UML models to OWL ontologies and then using them to generate mappings.

3. Develop a tool to generate mappings

   To complete this objective a tool is developed to implement the proposed approach of converting UML to OWL, generating mappings from it and provide a user interface to browse through the mappings, edit the generated mappings and provide users the ability to create manual mappings if required.

4. Evaluate the tool

   This objective will evaluate (1) the proposed conversion approach can sufficiently and correctly convert UML to OWL, (2) the tool can accurately generate and represent generated mappings and (3) the tool is easy to use to users with different expertise in UML, OWL and semantic mapping.

1.4 Contribution

The major expected contributions of this work are as follows.

1. Demonstrates the possibility of using UML for creating semantic mappings between software systems.

2. Exhibits UML to OWL conversion based on ODM 1.0 specifications and XMI 1.1 DTD.
3. Indicates the possibility of using business rules in automatic mapping generation.
4. Demonstrates a user interface for mapping creation system for use by system and knowledge engineers.

1.5 Technical Approach

As this research involves creating a tool to create semantic mappings using UML class diagrams initially a study of the state of the art in the areas of ontology mapping and matching techniques was done. Also a study of existing mapping tools was done to identify the strengths and weaknesses of the approaches used in those. Based on the analysis of the state of the art it was decided to design the tool to not only try to generate the mapping automatically by using techniques like structural matching but also use the users' domain expertise in generating and improving mappings and also providing the users with a suitable interface to verify, improve and add more mappings to the overall mappings. With these targets an initial design of the mapping generation strategy and the UML Semantic Mapping Representation Tool (USMRT) was created as a combination of two components. The First component generates the mappings and the second component is a user interface to represent the generated mappings in a way that is simple to understand and manipulate. For generating the mappings it was decided to first convert UML to OWL because of the similarities in the two languages, (Chan et al [Chan2001], Zhuoming et al [Zhuoming2012]), and standard specification on how one should be translated to another, [ODM 1.0]. Also, translating the model to an ontology language before generating mappings gives the ability to exploit the features that are available in the target language but are not available in UML, Staab et al [Staab2010]. To further understand the involved technologies and the current research a study of state of the art in ontology modeling using OWL, Ontology modeling with UML and generating OWL from UML was done along with the existing specifications on translating UML to OWL. For the user interface a web based interface was chosen so that users can access it remotely from their machine environments without the requirement for knowing about the tool itself.

Following the development of the tool experiments were performed to evaluate both the components of the tool. Experiments were also divided into two parts, each focusing on different aspects of the tool. First part of the experiment was done to evaluate the UML to OWL translation

and generated mappings and second to evaluate the usability of the user interface. The next section presents the overview of the evaluation.

1.6 Evaluation Overview

The evaluation is divided into two parts, each focusing on different aspects of the overall evaluation objectives. Experiments will mainly focus on the following aspects.

1. The proposed conversion approach can correctly and effectively convert UML to OWL and generate correct mappings.

   The evaluation of this aspect demonstrates that the proposed approach can convert the model expressed in UML class diagrams to OWL without losing or distorting the modelers design and generate correct mappings from it.

2. The implementation is easy to use and understand.

   This part of the evaluation includes two things, first that the use of tool using UML for generating mappings makes the process of creating mappings easier for people who do not have enough experience with semantic technologies and also for those who do and second to assess the usability of the user interface of the tool.

1.7 Thesis Overview

Chapter two presents a review of the state of the art in various components involved in this research, including ontology mapping and matching techniques, ontology modeling with OWL, ontology modeling with UML and OWL generation from UML with a focus on tools for automatic conversion. Chapter three presents the design of the system. Chapter four presents the technical details along with the implementation details of the tool. Chapter five presents the details of the experiments conducted for the evaluation of the tool along with the discussion on the analysis of results from the experiments. Chapter six presents the conclusion and future work followed by references and Appendices.

Chapter 2

State of the art

As the work of generating mappings automatically from UML class diagrams requires combining various different technologies such as ontology mapping and matching techniques, ontology modeling with UML and various other ontology techniques, this chapter covers the review of all the involved technologies and the various research in the corresponding fields. Figure 2.1 shows the graphical representation of the scope of this chapter.

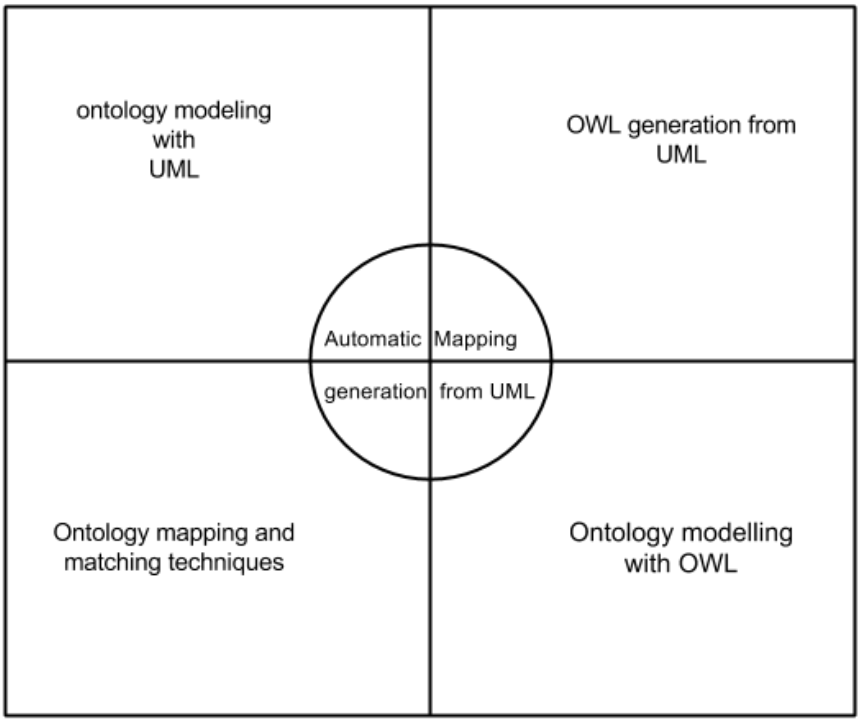

Figure 2.1: Scope of the State of Art

This review is organized as follows. Section 2.1 will discuss ontologies and ontology mapping along with the works that present various aspects of ontology mappings, techniques, methodologies and review of various mapping systems. Section 2.2 will present a discussion on OWL, ontology languages that will be used in producing mappings along with the various works on techniques using OWL for ontology modeling. Section 2.3 will have a short description of ontology modeling using Unified Modeling Language (UML) with the works on usage of UML for ontology modeling and finally in section 2.4 the generation of OWL from UML.

2.1 Ontology Mapping

Different definitions for ontology mapping can be found in literature, such as "formulae to find semantic relation between concepts in a model" Choi et al [Choi2006] or "process to find an entity in one ontology which has the same meaning in another" Madhvan et al [Madhavan2002] and the term could also be found used interchangeably with ontology matching depending upon the situation its being applied to and its scope. Here we define ontology mapping as a semantic integration process that tries to find correlation between components of different ontologies within the same or overlapping domains. There are other semantic integration processes like ontology merging, alignment, articulation and integration, which are often related to mapping but are different. For example, in ontology merging two different ontologies from the same domain are merged to generate a single ontology, whereas ontology mapping is used to find correspondence between ontologies from the same or overlapping domains and so could be used while merging. Various information sources like lexical, structural information or using a common reference ontology etc. and approaches like rule based, machine learning, graph analysis, probabilistic search have been used to discover mappings.

2.1.1 Ontology Mapping Problem

Choi et al [Choi2006] describe ontology as a specification of a shared perception as they are used to define concepts about data. Often ontologies for the same or overlapping domains are developed in different environments using different tools and processes, making them dissimilar in some aspects and as a result different parties working in these domains cannot understand each other's ontologies and because of the difference in the way the information is modeled, the same information is represented differently which causes similar information to appear different. This problem of systems not being able to understand each other is what ontology mapping tries to

solve. Creating mappings between different ontologies allows for them to be merged, integrated and aligned thus allowing for interoperability between systems created with different processes and further allowing information sharing between different representations solving the problem of information heterogeneity.

2.1.2 Mapping Approaches

The process of mapping generally consists of finding the correspondence between the two models and assigning scores to each of the found components and selecting the ones with scores above a certain threshold Choi et al [Choi2006]. Different approaches can be taken while creating mapping. One approach is to create a global ontology with features from individual ontologies showing how they map to each other and to the global one, then use the global ontology to map local ones as and when required. This approach is easy to implement and can be used in domains like data integration and knowledge management. The problem with this approach is that it needs a global ontology for it to work. The Second approach overcomes this problem to allow direct mappings between the source ontologies. Most implementations using this approach define an algorithm that compares some semantics like lexical or structural information between ontologies and use some rules to find mappings between them. This approach is more flexible and suits dynamic environments with changing ontologies like semantic web or evolving software systems, unlike the first where the global mappings need to be changed if any of the local ontologies change and so is more suited for a system that tries to generate mappings between software systems.

Ehrig et al [Ehrig2004] propose a technique for ontology mapping which combines a number of methods to calculate mapping candidates with different confidence values from which the actual mappings are calculated. To calculate the closeness values, different rules defined by domain experts are manually created and used by the system. The approach uses a number of methods like lexical comparison where the rule is that two things match if the labels assigned to them or their URI or their names match. Another method uses properties for comparison and the rule says that things map if they have the same properties and other such rules using restrictions, relations, instances, and transitive properties etc which are standard rules in ontology mapping. Additionally, they also use application specific rules which will be unique to every application and can give additional mapping candidates due to the fact that ontologies modeled for same application will have similar concepts and formats. All these rules are used in a hierarchical manner, starting with simplest to toughest and combine the prediction of each to get the final confidence value. To

combine the results of individual method a sigmoid function is used which allows giving higher weight to ones with high score and uses machine learning technique of neural networks to combine the results. To allow the sigmoid to work a threshold needs to be decided, the framework does this through a number of methods where they either keep a fixed value suggested by the domain experts or consider the highest value then subtract a predefined fixed value from it. The framework repeats the steps a number of times to improve the results and only considers best values from each iteration doing this also allows removing the duplicate mappings. It is important to note that not all methods are used in all the repetitions. Methods that give a large number of mappings are mostly repeated to reduce the number of possible candidates while the one that give precise results with high confidence value are not repeated.

The approach suggested here uses domain experts to define the rules of mapping. This can be useful in situations where domain experts are driving the mapping efforts as is the case with using UML for mapping generation. Another advantage is that as the model evolves only the rules will need to be changed without changing the implementation and since the human expertise is available can be done quickly and efficiently.

The techniques that consider the use the source ontologies directly without a global ontology use approaches like rule based, machine learning, graph analysis, probabilistic etc. to discover mappings. In rule based approaches various rules define similarities in the language or structure of the two source ontologies to get the result ontology. Machine learning approach as the name suggests, involves using machine learning techniques to discover mappings. The general approach is to use a number of learners, train them on the training data and then use the learners to find mappings. In Graph based approach, ontologies are considered as Graphs, and then a comparison is made between corresponding graphs and sub graphs to find the mappings. The next section discusses some tools that use the techniques mentioned above to explore which approach could better suit the task at hand in this dissertation.

2.1.3 Mapping Tools

According to Ngo et al [Ngo2013] every ontology matching system consists of some combination of following four main parts. The first part performs matching based on terminologies in the ontologies and various techniques like finding direct similarities can be used for calculation. Output of the first module is fed to the second module which uses the ontology structure to find similari-

ties. This module uses a number of techniques based on rules such as considering two ontologies identical if all their neighbors are identical. These rules are applied to the mappings discovered in first step and use the information to find more suitable mappings. Most of the techniques used in this module produce similar results that vary based on the input data and thus its performance depends upon the performance of the module used before it which gives it data to apply rules on. Third module uses symbols and constraints defined on ontologies to find mappings. This can take input from either step one directory or after step two and is generally used to improve the selected mappings from the earlier two modules. In this module the defined constraints are used to find clash between the candidates and decision is made to keep or discard them. A general approach is to use a threshold value and discarding everything that is below it. The last module selects the best from the discovered mappings. The performance of the overall matching system depends upon all the modules, but since the later modules use output from the once before them, they become more important and should be designed with more care with better techniques.

A number of tools using different approaches discussed in section 2.1.2 have been developed for the purpose of ontology mapping. Systems like LSD Doan et al [Doan2000] and GLUE Doan et al [Doan2002] use machine learning approach for ontology mapping.

LSD by Doan et al [Doan2000] is a system to produce one to one mappings between ontologies in same domain in a semi-automatic manner using machine learning techniques. LSD is based on typical machine learning technique where the system works in three stages. First where the system learns how to map from one schema to another using some training data and then the second step applies this learned knowledge to classify the test data to produce a number of mappings while assigning a certainty value to each classified value. In last stage the values are compared and highest certainty value is used to decide the final mapping. Like most systems that employ machine learning LSD also uses a number of learners, where each learner uses a different characteristic to learn and then use the same to classify. LSD also uses a Meta learner, which takes the results of all the learners and produce the final result from them using a stacking technique. LSD exploits characteristics like tags, number of words, lexical match and county name to train the classifiers. To train the classifiers LSD takes the manual supervised learning approach where they manually bring the data and then train the classifiers on it. To train, the system uses manually created mappings between the source ontology and a global one. To produce classification labels each learner is given the data it understands, i.e. the output of one learner is not fed to another, but each is given a slice of the test data and each of them produces a mapping value to it. All these

are then used to generate the best possible option using weighted sum.

GLUE developed by [Doan2002] considers taxonomies as the main component of ontologies and focuses on semi automatically finding the similarity between those. It uses machine learning techniques to discover mappings and use information in both data instances and taxonomic structure of ontologies for the same. GLUE uses a multi strategy learning approach where a number of base learners are created and the output of these learners is combined using a Meta learner which uses heuristics, domain constraints and other available information to fine tune the mappings. This extendibility of being able to include many learners in the learning chain is very useful to extend the system to a particular mapping use case and also allows for improving the efficiency by adding more learners with better constraints. GLUE uses two base learners, first a content learner which uses the frequency of words in text content of an instance and second a name learner that uses the full name of the instance which is the concatenation of all the names from the root to this instance. Meta learner in GLUE assigns weights to each of the base learners depending on the confidence in each and then uses the weighted sum to make the mapping prediction. GLUE uses Joint probability distribution functions to define similarity measure. GLUE also introduces the use of relaxation labeling technique in ontology mapping. It is a technique used to assign labels to nodes of a graph given a set of constraints which are of two types, domain independent and domain dependent. Domain independent constraints represent the interaction between the related nodes, example union constraint while domain dependent relates to interaction between specific nodes. GLUE consists of a distribution estimator, Similarity estimator and relaxation labeler. Distribution estimator takes two taxonomies and their data instances as input and calculates joint probability distributions for each concept pair. These are then fed to the similarity estimator which produces a similarity matrix. Finally, the relaxation labeler finds the best mappings using the specified constraints. Application of relaxation labeling to ontology mapping also appears to be a good approach. It is an already proved matching technique used in computer vision and Natural language processing and so, as the authors show suit well to ontology mapping.

Both the systems discussed above highlight some important aspects of machine learning approach. First, the requirement of quality training data which makes this approach difficult to use in cases where training data is not available or the source models change overtime requiring changes to any training data that might be available. This makes the approach less suitable for extracting mappings between software models where training data is difficult to obtain and where the systems continue to evolve over time. Second, this technique requires multiple iterations and other steps

to extract good results from all making it slow.

Systems like Anchor-PROMPT Noy et al (Noy2001) are based on the graph approach of ontology mapping. It's a tool that considers the concepts in a source ontology as the nodes and relations between these concepts as the labels of a directed graph. To find mappings the system navigates the graph and assigns a likeliness score to the nodes. To calculate the score the system makes an assumption that in a directed graph if two connected nodes are similar then there exists a possibility that the nodes in the path of these two connecting nodes are also similar. To start the comparison the system takes a set of related elements called anchors (hence the name Anchor-PROMPT) which are often set manually by the users or can also be identified automatically. It then navigates the graph, finding similar concepts assigning scores to them and finally accumulating the scores. The scores are then used to output the sets of mapped concepts. The evaluation shows that the performance of anchor-prompt decreases with increasing size of the source ontologies and it also performs poorly where the source ontologies have significant structural differences like one of the source ontologies consisting of deep hierarchies within a concept while other one with shallow hierarchies. Here we can see that the graph approach is more suitable for cases where the source ontologies are structurally similar and the speed of conversion is not of major concern. In case of creating mapping between the software models done to achieve integration the source models may vary structurally and so this approach doesn't quite fit the need.

Systems like PROMPT Noy et al [Noy 2000] and Quick Ontology Mapping (QOM) Staab et al [Staab2004] use rule based approach to find the mappings between ontology models.

PROMPT by Noy et al [Noy 2000] is a tool for semi-automatic merging of ontologies. PROMPT does not try to generate mappings by itself but helps users to create them by suggesting them the possibilities. PROMPT uses both lexical and structural similarities of the source ontologies to derive the suggestions. It also responds to the user actions by adding them to the equation. Whenever a user performs an action PROMT takes the result of the action and reruns the rules to come up with new suggestions and conflicts and prompts them to user so that they can be used in next steps. PROMPT only measure one resemblance which is the equality of tags and nothing else, it's fast and simple and tests show that it produces up to 74% correct suggestions. One of the limitations of PROMPT is that it only uses exact label matching and so cannot map concepts that do not have the same name.

Staab et al [Staab2004] developed QOM which is a software that uses the same steps for finding mappings as most algorithms using lexical analysis but optimizes some of them to get the desired efficiency. First optimization is in the search selection step where it uses dynamic programming approach to divide mapping candidates into categories to be considered immediately, adding them to be looked at during further iterations, and ignoring some using a defined threshold. Second optimization is in the way similarity is computed. Most approaches use pair wise comparison while QOM does an n incomplete top down approach. First two optimizations reduce the candidate mappings and comparison time and the final one reduces the number of iterations which is just repeating all the algorithm steps to refine mappings. QOM does this first on lexical knowledge and then on structure and shows that maximum iterations required was 10 irrespective of ontology size. By implementing QOM the authors try to show that the mapping efficiency can be significantly improved by slightly sacrificing effectiveness and show that the effect of reduced efficiency is marginal compared the gain in efficiency. In most algorithms time to produce the overall mappings is proportional to the number of individual mapping candidates and number of iterations over the model to find those. So, the efficiency of an algorithm can be improved by limiting the candidates and iterations, this is the idea behind the presented approach where they try to reduce the candidate mappings and iterations to improve efficiency and at the same time keep the error to an acceptable level.

Summary

This section discussed a number of approaches to solve the ontology mapping problem by using information provided by the ontologies and methods that work with this information to best estimate the mappings. Table 2.1 shows the comparison of 3 widely used mapping approaches and table 2.2 shows the comparison of automatic mapping generation tools comparing aspects most important from the perspective of this research. Comparing the approaches and the tools

Table 2.1: Comparison of Mapping Approaches.

Rule Based Approach	Machine Learning Approach	Graph Based Approach
Uses user knowledge in the form of rules to create mappings.	Users are mainly involved in creating training data or improving the generated mappings	Users are involved in improving mappings.
Doesnot need training data	NeedsTraining data	Does not need training
Fasterthan then other two	Generallyfaster than graph based but slower than rule based	Slowest,among all three
Mapping performance largely depends on the rules provided by the user.	Performance depends on the quality of training data and can improve overtime for similar domain models.	Performance,depends on the size of the model and is particularly affected if the models,differ in structure.
Suitable where the experts are available to provide rules and requires fast mappings	Suitable for situations where training data is abundant and the process needs to be,repeated for same domain.	Suitable,for situations where the models are similar to each other in structure.

Table 2.2: Comparison of Mapping Tools

Tools	Approach	User Involvement	Knowledge types used	Input	Output	GUI
PROMPT	Rule Based	Post Creation	Structural, lexical	2 source Ontologies	1 merged ontology	No
QOM	Rule Based	Post Creation	Structural	2 source Ontologies	Result ontology	No
LSD	Machine Learning	Pre & Post Creation	Instance, domain, lexical	Schemas & their instances	Mapped terms	No
GLUE	Machine Learning	Pre & Post Creation	Instance, domain, lexical	Taxonomies & ontology instances	Mapped terms	No
ANCHOR-PROMPT	Graph Based	Pre & Post Creation	Structural	2 sourceOntologies	Mapped terms	No

supported the theory that rule based approach better suit the scenario at hand because the target audience, the domain experts can provide expert rules for good mapping performance but the two source software models to be used for mapping generation may not have similar structure making graph based approach less suitable also lack training data for different models which evolve continuously makes machine learning approach difficult and less suitable. These also helped identifying some requirements for the tool design, for example, most existing rule based tools use user expertise after the mapping process. Here we can use them both before and after the mapping generation. Second, the tools generally use structural and/or lexical rules which are good as basic rules to find the mapping candidates and so it was decided to keep similar rules in built in the system and also allow the users to specify them. Also, it was found that most tools do not provide a complete interface and require help from other tools for it. This limits the capabilities of the tool making it dependent on other tools. For this reason it was decided that tool should have a user interface that not only allows browsing through the model, but also provide another feature like the ability to edit and query model making it more feature rich.

2.2 Ontology modeling techniques with OWL

OWL the Web ontology language is a modeling language to model ontologies using RDF triples and to extract and process the content of information. It is the latest part of w3c recommendations for semantic web along with XML, XMLNS, RDF and RDFS and adds more terminology for describing ontologies making it more expressive in the description of ontologies authored using it. OWL is based on description logic and defines classes, properties and instances in an RDF docu-

ment. A class is a representation of a group whose members have common properties. Properties define relationships and are of two types, object properties that define relations between resources and datatype properties which define relation between resource and its data. Instances are the members of class. OWL is further divided into following three sublanguages depending on the complexity, scope of use and with increasing expressivity.

1. OWL Lite

   It is the smallest subset of the complete OWL specifications, has lower complexities and a small number of constructs for modeling. It should be used if one has simple modeling needs.

2. OWL DL

   This has the full expressivity, but with some restrictions, for example a resource cannot be a class and instance at the same time.

3. OWL Full

   OWL FULL is same as OWL DL without any restrictions. This should be used if one needs all the features.

Since all the modeling in OWL is in the form of RDF triples, changing any model only requires changes to a particular triple without affecting any other this makes the model very flexible which is important for ontologies. OWL could be used for reasoning over the data and OWL profiles allows choosing of different types of reasoning for different performance requirement making it more efficient. Below is the discussion of some approaches that enhance OWL's modeling capabilities to implement models.

Motik et al [Motik2009] propose an extension to OWL using description logic and first order rules to increase OWL'S capabilities of expressing the domain knowledge being modeled and representing the relations between its objects while still allowing the use of OWL primitives for modeling the parts which do not require additional features and capabilities. OWL languages have been designed to permit a higher and better level of analysis over the designed model while sacrificing some of the articulation capabilities. Due to this most of the applications using OWL either

become too complex and verbose trying to express the domain completely or do not do it sufficiently. This problem with OWL of reduced details in expressing ontologies of complex data can be mitigated by following extensions to the language. First, using directed labeled graphs similar to description graphs to represent concepts with nodes of the graph and rules by links between them. The second extension is to provide rules that allow making subsets of a graph more specific for particular set of functionalities. Next it allows specifying the correlation between these specialized subsets with a particular graph sequences so that they can have partial overlapping concepts. These extensions can only be applied to OWL when modeling a domain with finite number of concepts and instance which is the case with most and so these can be applied to most. The main problem the recommended extensions try to solve is to increase the expressive powers of OWL while maintaining the ability to derive inferences from the data sufficiently high.

Bouquet et al [Bouquet2003] propose an extension to OWL called contexted web ontology language COWL to allow localizing information in ontologies which is otherwise not possible with standard OWL syntax. Since OWL does not define mappings, they are performed by importing the complete model to be mapped and then creating mappings, this makes all parts of ontologies to be accessible to the other model mainly because ontologies modeled using OWL do not have a concept of private non-sharable regions. This may not always be desirable, for example situations where different clients accessing a system have different access rights and are allowed to query only the parts that they have access to. The proposed solution tries to overcome this limitation by introducing the concept of context to ontologies. This allows defining which ontologies are local and does not have outside access and which are global with free access. This is specified by the syntactic extensions to OWL making only global parts to be shared when someone imports the ontology model. Another major limitation of OWL that the proposed framework tries to remove is the lack of directionality concept in OWL which means that if a concept in the ontology “A” maps to some concept “B” in other ontologies then all the individuals of "B" also map to all individuals of "A". Again, this may not always be desirable. There may be cases where one wants to map from a less general ontology to more where the above mentioned condition will cause problems. COWL tries to solve this by specifying rules that decide the directionality of the mapping. Final contribution COWL makes to improve OWL is to allow adding context to mappings in OWL which means allowing different ontologies to interpret each other by knowing which concepts in two ontologies refer to the same thing. Again, this is done by specify rules adding more semantics to the standard bag of semantics available in OWL.

Summary

This section discussed OWL and its three variants and also discussed a few approaches that can help overcome the limitations of OWL one can face while expressing software systems in it. The flexibility and expressivity of OWL and its ability to be tailored for a particular use case makes it a good modeling language for the semantic web and developing ontologies that can be shared and reasoned on. This study helped deciding OWL DL as the target ontology language for the system.

2.3 Ontology modeling and development using UML

UML the Unified modeling language is a well-known, standard modeling language for modeling software in object oriented systems. It is used for graphical modeling of the system allowing for structural and behavioral modeling. Structural modeling is done using diagrams like class diagram, object diagram, profile diagram, component diagram etc. representing objects, relations between them, their attributes and operations allowed on them. Behavioral modeling is done using diagrams like sequence diagrams, activity diagrams that show how different objects collaborate and state machine which show their internal state. For specifying constraints about objects in the model UML uses OCL (object constraint language) a formal language developed to be used by system modelers to specify rules like invariants on classes and types in the class model, pre and post conditions on Operations and Methods, constraints on operations, and other such rules on models. OCL can also be used as a query language to query the model. UML is a graphical representation of model designed for easy human understanding. In order to allow for exchange of this between machines XMI (XML Metadata Interchange) format which joins XML, UML and MOF are used. This makes UML machine understandable thus allowing automatic exchange of models between software systems that can consume it. MOF (Meta Object Family) is a meta modeling standard used to define UML models. Thus when exchanging XMI the metadata is also exchanged which allows the systems to understand the model and reason over it. UML is used to model concepts and stores related data and rules with it. This is similar to ontology development and ontology development can benefit from all these features of UML. It can be used to graphically model ontologies, there are a number of mature UML tools that can be used and since it is widely used by software engineers their expertise could be used for knowledge modeling of systems. However, there are some inherent differences in the way UML allows modeling compared to ontology languages like RDF, OWL etc. For example, in UML properties cannot be modeled

individually, they are only in association with a class, whereas in OWL/RDF they can be. Because of this difference either a mapping or extension to UML are required for doing it. A number of researchers have tried to use UML for ontology development and modeling. Some of such works are discussed below.

Baclawski et al [Baclawski2001] discuss similarities and differences between UML and ontology language RDF and DAML+OIL and provides some recommendations to extend the UML Meta model in order to allow UML usage for ontology development. According to the authors UML features like profiles, global modularity, extension mechanisms, visual modeling etc. could help ontology development and as these are not present in most of the ontology languages, UML could be used instead. To show the similarity between UML and knowledge representation (KR) language DAML+OIL a mapping is presented assuming that the UML class diagrams were created for DAML+OIL modeling and for each feature of DAML+OIL not supported by UML an extension is proposed to UML to support it. DAML+OIL SubClassof property is mapped to UML specialization/generalization i.e. subclass/superclass relationship. SubPropertyof is mapped to specialization of association i.e. a labeled association between UML classes and other such similarities that can be directly mapped are given. A number of incompatibilities between UML and KR languages are identified, First, the notion of monotonicity. A system is monotonic if adding new facts cannot falsify old facts. All the KR languages are monotonic while UML are not and so are inherently different. Second, UML has a well-defined separation of Meta levels while KR languages do not. So, it becomes ambiguous while mapping from one to another. Third, a KR language does not have any modularity mechanism like profiles, which makes it difficult to classify which ontology a resource is a part of. KR language property notion maps to UML's association, a KR property can be modeled on its own but a UML association cannot be. The authors suggest adding more elements to UML Meta model to address the incompatibilities and make it suitable for Knowledge representation and ontology development. Some of the recommendations are, first adding a metaclass property to UML Meta model and make it a first class modeling element. Second, adding Meta associations like unionof, intersectoinof, complementof and allow creating of classifiers from other classifiers. Finally, as suggested deciding on the purpose of mapping beforehand is very important because different mapping requirements might call for different approaches and one may not be suitable for all. For example a mapping that needs to be consistent may not necessarily be have to be correct and so exact translation might not be required. This suggests that extension to UML may not be required for all cases of mappings, but for some that absolutely require the incompatible parts. This work presents useful insights on the differences between UML

and ontology language and how one can overcome them and helped in understanding the limitations of UML as an ontology language.

Chan et al [Chan2001] Investigate use of UML in ontology modeling and as a tool for mapping from knowledge models to software models and apply the modeling technique to petroleum remediation domain. The authors argue that knowledge modeling is important in software development as it can be used for a higher level of knowledge reuse than just software reuse. However, knowledge modeling alone cannot be used to produce good software because it's related to user centered stage of development process and lacks system centered support. To overcome this limitation and allow better use of knowledge modeling in software development process authors propose using UML which is a standard language used for modeling software systems along with OCL used to specify constraints on models in UML to model ontologies. Modeling the same concepts in both ontologies and UML they show how the concepts match each other in both formats.

We et al [We2003] discuss the use of ontologies in multi agent systems and using UML to design their ontologies. In multi agent systems, agents built using different technologies need to communicate with each other and understand each other. This requires them to know each other's model and reason about that model in real time. Ontologies are used for defining specifications of a domain and so these requirements can be fulfilled easily and efficiently by using ontologies to model the domains and environment these agents work in and help the agents to work better by being able to query the domain knowledge with data in a uniform manner. Since, most of these agents and domains are developed using object oriented architecture and model using UML, their UML diagrams already have all the domain knowledge in them and can be used to generate ontologies from existing knowledge. UML alone cannot be used to model ontologies because of lack precise semantics and constraints between entities and so OCL (Object Constraint Language) is used along with it for generating ontologies. To be able to use UML successfully and usefully for ontology modeling it should be able to model the domain in ontology semantics and also allow automated reasoning over that model. To model ontologies with UML they propose a set of rules to extend the UML class diagram enabling the ontology semantics in it thus allowing for a broader range of ontology semantics modeling. These rules allow modeling at the agent level supporting communication between them using relations between them as predicates. UML does allow reasoning over the modeled domain, but it is limited and cannot be done automatically, to overcome this limitation, certain rules are proposed which could be used for analysis and transforming class diagram to allow reasoning. These rules are applied in a series where each rule transforms the

diagram by certain extent until finally the relation for reasoning can be deduced.

Staab et al [Staab2010] developed a framework called TwoUse that allows modeling of software systems using the features of both UML and OWL together, thus allowing to benefit from the modeling powers of both and overcome limitations of either of the individual modeling languages. The proposed framework tries to integrate both in modeling so that good features that are available in either can be used. For example OWL allows for specifying specialization, relationships etc. dynamically. Which is good as it can be updated dynamically as the model evolves. UML on the other hand only allows these to be static and so if the model is based on OWL it would be better. UML has advantages over OWL such as unlike OWL, UML allows to model dynamic behavior. So in this case UML model is better. The framework tries to combine such advantages of both in modeling by integrating meta object family (MOF) meta model of both, allowing reasoning over it by using OCLDL and creating a hybrid model. The framework consists of four parts. First, logical syntax that combines the MOF meta model of both UML and OWL creating a common underlying structure, transforming the individual meta model to twouse meta model. Second, the actual syntax based purely on UML along with text syntax to write OWL description into it, which creates a form of hybrid diagram containing both UML class diagram and Owl class diagram with logical views. Third, the rules to transform from OWL to UML for semantic integration and Last OCLDL syntax for writing queries and reasoning, this part has some limitations like queries might result in exceptions because they return classes that have not been translated into twouse meta model and the developers need to take care of such situations. Unlike other approaches dealing with integrating OWL and UML the use of this framework requires the users to understand both the technologies involved.

Summary

This section discussed UML features and how it can be used in ontology modeling and also discussed works that try to use UML for this purpose. From the existing literature it was found that most work try to use UML for ontology modeling by using its graphical powers or extending it to allow ontology modeling. These approaches can only be used with the new models and not the with old ones which limits their use. This work tries to find a way by which knowledge from existing models can also be extracted.

2.4 generating OWL from UML

UML to OWL mapping algorithms can be divided into two different classes. First, where the focus is to let OWL ontologies be modeled by using UML based visual tools. This approach has a number of problems like the ontology developers need to know UML tools which may not be a case always and not being able to extract ontologies from existing UML's. The second approach focuses on letting the software engineers model the domain knowledge in UML and then extract this knowledge from UML and convert it to OWL by using correspondence between the two languages. This second approach is used by Zhuoming et al [Zhuoming2012] to automatically generating OWL ontologies from UML while preserving most of the semantic information encoded in UML class diagrams. A number of features from UML and OWL have correspondence with each other and can be mapped directly while there are some features that exist in one and not in others. Presented approach only considers part of UML that are absolutely necessary to build ontologies. The algorithm first translates the UML class diagram using description logic and then converts it into OWL DL which was designed to support the existing description Logic and uses description logic to specify meaning. The algorithm follows a standard process of mapping generation and conversion is performed using the constructed mapping rules in a two-step process where first the UML symbols are translated to OWL identifiers. This step creates corresponding OWL identifiers for UML symbols like owl:class for uml class, owl:property for uml class attribute etc. thus first creating an owl structure which is then filled up with details in second step. Second step converts UML elements to OWL axioms. Conversion is done using the specified mapping rules in order of specificity to generic, starting with class axioms followed by association, then roll and finally generalizations, these values are then assigned to identifiers created in the first step. The approach presented here is somewhat limited and may not preserve all the semantics. It only considers the core parts of UML that has a direct correspondence to OWL and not once which have none. This is not a good idea for a mapping algorithm because it will miss the important domain knowledge that cannot be represented by simple core specifications of UML and result in a mapping that does not represent the modeled domain sufficiently.

Another approach to extract OWL from UML is to use Extensible Stylesheet Language Transformation (XSLT). Gasevic et al [Gasevic2004] present an approach that uses this technique to automatically generate OWL in XML format from UML using Extensible Stylesheet Language Transformation (XSLT) and OMG provided specifications like Model Driven Architecture (MDA), Ontology Definition Metamodel (ODM) and Ontology UML profile (OUP).

The approach defines a framework that uses ODM for ontology language, OUP in XML Metadata Interchange (XMI) format and produces bidirectional mappings between them and converts XMI to OWL XML. This approach exploits OUP provided extensions like tag definitions, stereotypes etc. In order to allow UML to represent broader ontological semantics that can be mapped to OWL semantics. This allows for defining ontology equivalent of UML concepts that do not have an Ontology correspondence by default. Their idea is to use existing UML tools that support OUP to export UML in XMI format and produce an OWL XML document from it. Since both source and destination formats are XML, source can easily be converted to destination using XSLT which is a standard technique for producing and converting XML documents in different formats. They define an XSLT document with all the rules of transformation according to OUP and ODM and feed this document along with source XMI document to an XSLT processor, which then applies the rules on source and produces an OWL XML document.

The idea of using XSLT to perform transformations is good because both source and destination formats are XML and this is a well-accepted, simple and standard way of manipulating XML documents. But this also limits the use of this approach and its suitability for other situations where output is required not to be XML but something else like and OWL N3 or turtle syntax. Another good thing is that this approach makes use of all the existing standard, languages and tools to achieve mappings in both cases where direct mappings are possible and where the concepts don't overlap without suggesting any extensions or changes to anything, this makes it easier and more acceptable to use. Another limitation of the current approach is its ability to produce only one to one mappings.

Summary

This section discussed two approaches for automatically converting UML to OWL and the pros and cons of each. Table 2.3 presents the summary. The comparison of the above two approaches allowed identifying the strengths and weaknesses of each and helped in deciding the final approach. From the comparison, it can be seen that none of the approach is suited perfectly for the task at hand and so an approach combining the two was taken. The conversion approach used uses java to convert UML to OWL but unlike the approach presented above it does not use a fixed model created by the designers of the tool instead in but uses the XMI 1.1 schema to create the model and then uses ODM 1.0 specifications like the second approach to convert the UML objects

Table 2.3: Summary of UML to OWL conversion approaches.

UML to OWL conversion approach	XSLT Based	Java and Description Logic Based
Advantages	Simple, Conforming to standards, Broad range of conversion.	Suitable for multiple formats like XML, N3 etc., Complex Mappings Possible
Disadvantages	Only One to one conversion, Not Suitable for formats other than XML	Complex, not standard, conversion only possible where direct correspondence exists.

to OWL objects using JENA.

2.5 Summary

This chapter first discussed the state of the art in various technologies involved in the project. First, ontology mappings approaches and tools with an aim to identify which approach best suits to fulfill the research goals and it suggested that rule based approach is the one to go for. Second, ontology modeling with OWL to identify to what extent OWL can be used to express mapping between software models which helped in finalizing OWL DL and also understanding the techniques that can be used to extend OWL to express things default OWL cannot. Third, was the review of ontology modeling with UML to establish the current state of research where UML is being used for ontology development and finally on techniques of converting UML to OWL which helped deciding the approach using Java, JENA, ODM 1.0 specification and XMI 1.1.

Chapter 3

UML Semantic Mapping Representation Tool (USMRT) Design

3.1 Overview

Unified Modeling Language (UML) is standard and widely used technology for modeling domains, applications and business processes. UML's features like ability to model graphically and its maturity as a modeling framework along with the availability of a large number of experts due to its widespread use make it a good candidate for use in knowledge modeling. The growing interest in using UML for ontology modeling is evident from the fact the Object Management Group (OMG) has developed standard Ontology Definition Metamodel (ODM 1.0) specifications for ontology modeling using UML and there have also been research in developing,(Baclawski et al[2001]) and generating,(Gasevic et al [Gasevic2004]), ontologies using UML. The system designed in this research tries to make using UML for semantic mapping generation easier and faster for domain-aware users by providing them an easy to use interface and a way to input their domain knowledge into a system to generate mappings.

This chapter discusses various aspects involved in the system design starting with outlining the requirement for the tool followed by discussion of functional architecture and each of its components next the discussion on the design of various components in the system along with the design decisions, rationale behind those decisions and challenges.

3.2 Requirements

The development of USMRT tool includes developing an approach to automatically generate semantic mappings from UML class diagrams. Since completely automatic generation of mappings is still not considered possible second important part of the development is to develop a user interface that allows users to check the generated mappings, edit the mappings and most importantly create additional mappings manually with the tool. The initial design targets support for generating simple, direct mapping types and complex direct mapping types that can then be extended to include other mapping types. The following main requirements were identified for the design.

1. Converting the data models in UML to knowledge models.
2. Finding the mappings between the two models.
3. Displaying the generated mappings.
4. Updating the models with new and edited mappings from user.

Apart from the primary requirements mentioned above the secondary requirements for the tool were to

1. Allow querying the generated model using SPARQL.
2. Allow downloading the model with mapping.
3. Allow Users to upload UML models to the server.

3.3 Functional Architecture

Based on the requirements identified above the system functions were divided into four main parts. First the User Interface which deals with all the user interactions, including UML model upload, ontology graph and mapping display, graph editor and mapping creation. Second function is to convert the model from UML to OWL. Third function is to generate mappings and forth function to update the mappings in response to user updates. All the parts share the UML and OWL models to complete their functionality. Figure 3.1 shows the functional architecture of the tool.

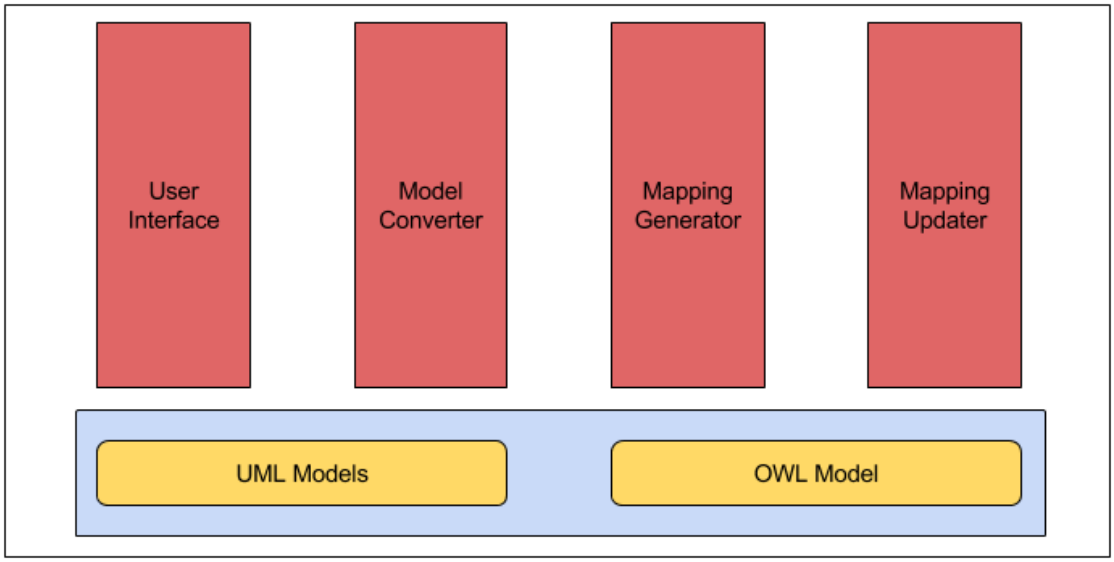

Figure 3.1: USMRT Functional Architecture

3.4 Design

Figure 3.2 shows the approach taken in designing the tool. The tool takes two UML models and user rules as input. First the UML models are converted to OWL model. The system then applies the rules to it and generates the mappings which are sent to the user interface in JSON format for displaying. Finally the users are involved who can update the mappings manually on the screen or change the rules and start the process again to generate the mappings by the system.

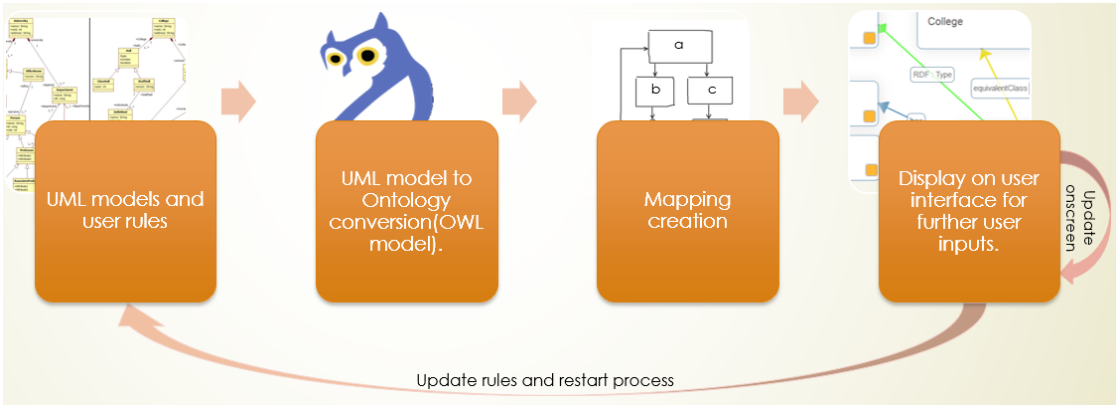

Figure 3.2: USMRT approach

The first design decision was to separate the user interface from the rest of the system and follow client server design with Model View Controller (MVC) architecture. This allows for

the client to be a thin client without being tightly coupled to the server and can be developed separately, making sure that once the client is designed and the communication protocol between the client and server remains same any change to the server code and logic will not force the client change thus keeping the users shielded from any change. The communication between client and server is based on Representational State Transfer (REST) architecture according to which each resource on the server is identified by a unique URI, in this case it is the model resources, the query url, model update etc. This enables the client to be completely decoupled from the server. The data transfer between the server and client was decided to be in JavaScript Object Notion (JSON) format because the OWL model can be easily produced in JSON and JSON is easy to manipulate on the client side using JavaScript.

3.4.1 Client Application Design

The Client was designed as a single page web application that the users can run from their browsers. A web based interface was decided because it offers more flexibility by not asking the users to install a new application and can be accessed from anywhere. A single page design was chosen because it reduces the efforts required for users to remember where they are in the application and knowing how to navigate to different parts of the application. Figure 3.3 shows the overall client application interface.

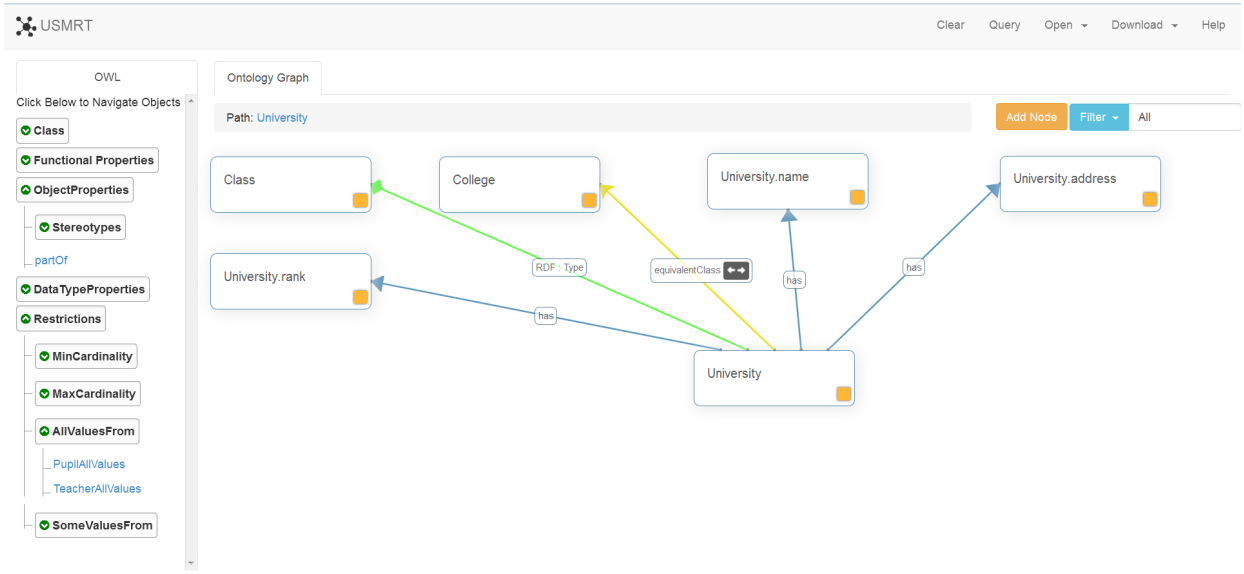

Figure 3.3: Client Application Interface

The important challenges to address while designing the client were.

1. It should be easy to find the data that user is interested in.

   Since there could be a number of mappings and a lot of other details associated with them, the interface should be such that it does not clutter and allows the user to find the things they are interested in. This was addressed by designing filters which the users can use to filter out the required information on display. After testing different filter designs two levels of filters were decided. The first filter is in code which groups the model elements according to their types and the displays them as a tree. Figure 3.4.a shows the groups and tree structure.

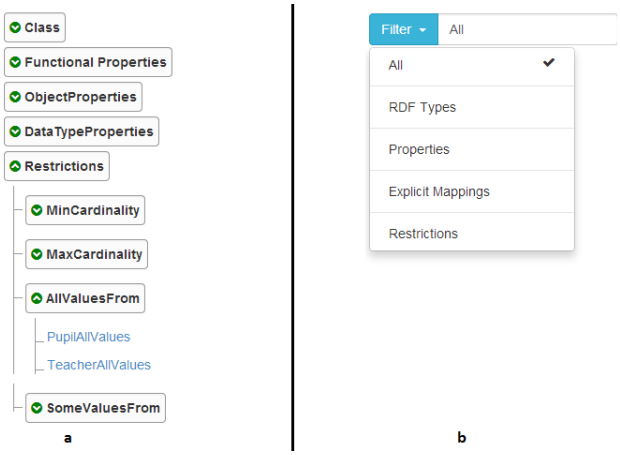

Figure 3.4: Data filters on User Interface

   Second Filter allows users to select the type of view they want to see for example explicit mappings, properties etc. Figure 3.4.b shows the user selectable filters.

2. The mappings should be displayed in a manner that is easy to navigate, understand and relate to UML models.

   This is one of the most important design issues for the user interface. A number of possibilities like displaying a tree structure, table and graph were considered and finally graph display was decided where each of the nodes in the OWL model is represented by the graph node and graph edges between the nodes representing the relationships and mappings between the nodes. Figure 3.5 shows the graph display of the university node in the OWL model. The users can navigate the graph by clicking on the nodes to go deeper in the graph moving on the next nodes and to return back click on the breadcrumb path to go back. This reduces the amount of information in one view making it easy to comprehend and easy

to navigate. The graph display also makes it visually similar to the UML class diagrams, making it easier to relate between the two.

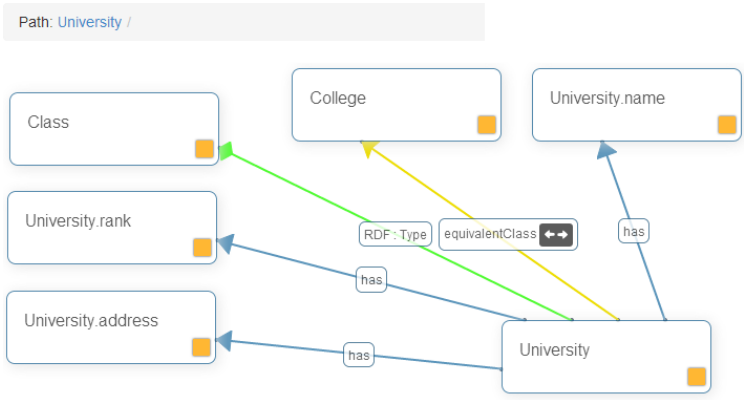

Figure 3.5: Graph display

3. The interface should make editing and creating mappings easy.

Another requirement for the tool is to be able to edit and create mappings. The challenge here is to make it easier and in the context of the node which the user is viewing at any given time. This challenge was solved with design by allowing edit by clicking on the link representing the relation or mapping. Figure 3.6 shows the design for editing.

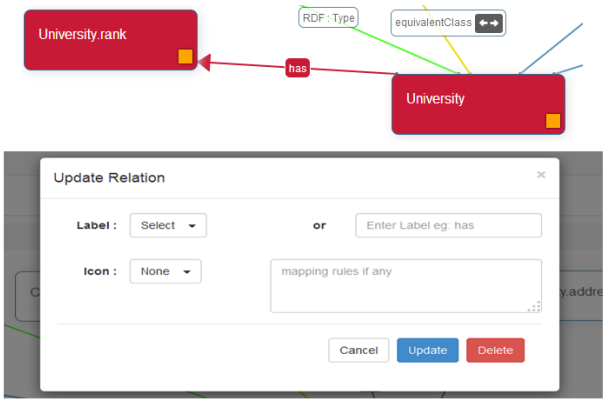

Figure 3.6: Mapping Editing Screen

In order to keep the design uniform, the process of adding new mappings was kept similar to the editing process. The design makes sure that the users can only add new nodes and

mappings to the node that is on the main display. Figure 3.7 shows the process of adding new mapping.

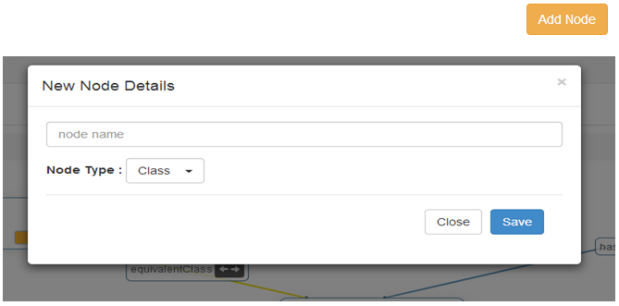

Figure 3.7: New Mapping Creation Screen

For the secondary features of the client a menu design was chosen in which the features are grouped together and put in a menu for easy access. This design was chosen because these features are less frequently used and putting them in a menu at the top of page make them non-intrusive yet easily accessible. Figure 3.8 shows the secondary features menu.

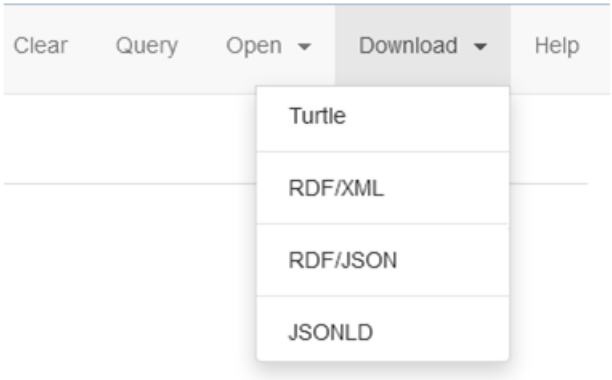

Figure 3.8: Client Secondary Features Menu

3.4.2 Server Application Design

The server application was designed as a Java web application. The application is designed as a collection of services with REST endpoints that the client connects to. Doing so allows the client to access the different functionalities as resources and be decoupled from the server. Major challenges to address in the server design were as follows.

1. Getting the information from data domain to knowledge domain correctly.

   The input to the tool is a pair of UML models and the goal is to extract semantic mappings between the two. In order to extract the mappings it was decided to first convert the data model in UML to knowledge model in OWL as soon as possible and then try to work in the knowledge domain to extract mappings from them. This technique gives the ability to use features of tools like JENA and its inference capabilities to mine mappings. For this purpose different techniques from the state of art were looked at and different designs were tried and tested to evaluate the strengths and weaknesses and suitability of each for the task at hand and then the used approach was finalized. The first approach tried was to use an Extensible Stylesheet Language (XSL) stylesheet and Extensible Stylesheet Language Transformations (XSLT) to transform a UML model in XMI (XML Metadata Interchange) format to OWL in RDF/XML. This approach was easy to use and implement, but is less flexible. It requires changes to the stylesheet for different models and domains and was not able to generate anything more than one to one mappings. The second approach was to convert the XML to Java objects using XML parsing in Java and then convert the Java objects to OWL model using a template model that was already created in the code. Though this approach better than the XSLT approach in terms of flexibility had the problem of the model being translated to internal model which not be portable to other tools. Further, it required developers of the tool if any change was required for the model. Finally the approach that was chosen was the approach to translate Java objects to OWL model on the fly without a template. This was done using the rules of ODM 1.0 specifications XMI schema, making it standard and portable.

2. Generating Mappings between the models.

   This is another important aspect of the system and a number of possibilities were considered in designing this. Most current approaches that try to generate mappings between knowledge models use things like matching lexical or structural similarities between then models. Assign some scores to the matched elements and use some threshold for them to say that to concepts map to each other if they have a matching score above the threshold, (Choi et al [Choi2006]). The approach designed in this work also uses the structure to find relations and mappings on the base level which can be directly inferred from the models. But it mainly uses rules to generate mappings. The system defines a set of default rules which

are executed on the models to draw mappings, unlike most other approaches the system tries to exploit the fact that the domain experts, design UML models with some rules and constraints and that the same rules can be used to generate mappings between the models in the same domain. To do this the system design allows the domain experts to provide rules in a standard way using decision tables or domain specific languages which are then converted to a format used internally by the tool. These rules are then executed on the models to create mappings. The rules themselves are not the part of mappings, but are simply used to generate mappings.

3. Getting the mappings in a format that could be used by the client for displaying.

   The challenge here is to get mappings in a format that is suitable for different users to understand as well as easy to display. To do this it was decided that the mappings and model would be in OWL DL, but the format was not fixed and option was kept to let users decide the format they wanted and download what they required. Since the client display creates a graph using HTML and JavaScript the transfer from server is in RDF/JSON format because it is easy to manipulate with JavaScript.

3.5 Summary

This chapter discussed the design requirements of the USMRT tool along with design challenges and the decision that were taken to fulfill those requirements. The next chapter will discuss the implementation of design discussed in this chapter.

Chapter 4

Implementation

This chapter discusses the implementation details of the system and will also include details of the libraries and frameworks used. The code for the implementation can be found in the accompanying DVD. Figure 4.1 shows the technical architecture of the system

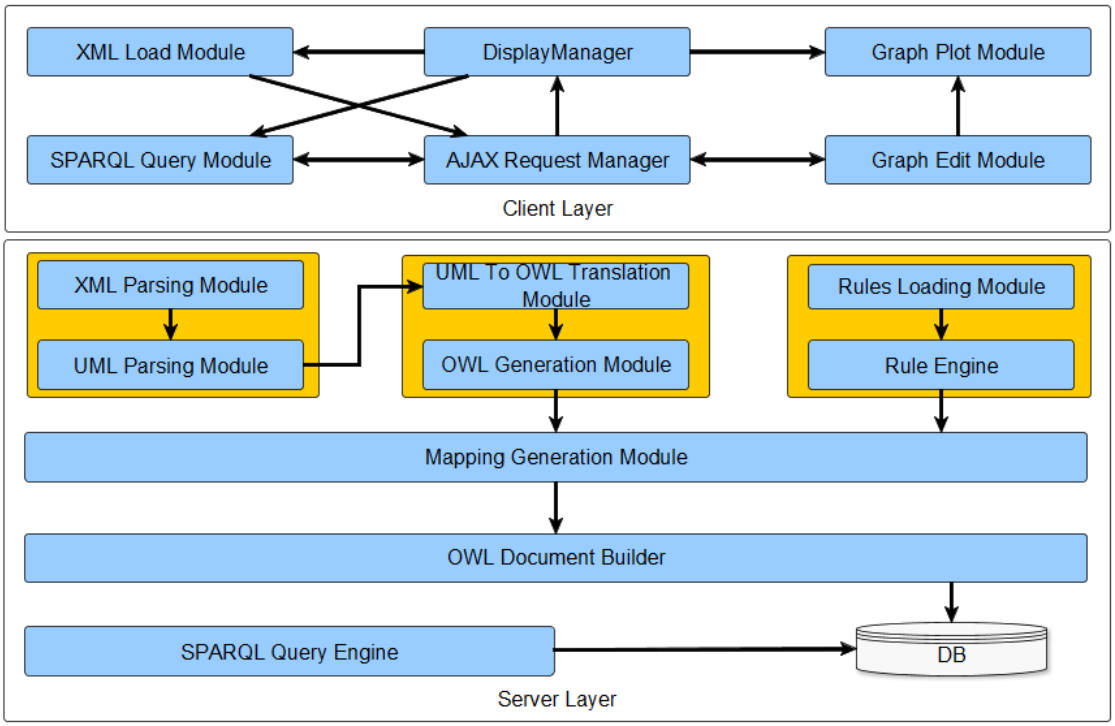

Figure 4.1: USMRT Technical Architecture

4.1 Client Implementation

The client implemented as a web application allows for uploading the UML model in XMI format to the server, displays the mappings, allows user to edit and add new mappings and execute SPARQL queries on the generated model.

User interface is implemented using HTML 5, CSS, JavaScript and Ajax. The HTML page is developed in HTML5 and all the client side programming is done in JavaScript. Apart from the core HTML and JavaScript following libraries were used for the features they provide.

1. Bootstrap: It's a HTML, CSS and JavaScript framework for developing responsive pages. It was used because it provides a number of ready to use components that can be used to create HTML pages.

2. Font Awesome: It's an icon bundle to be used on the pages. Used in the project for page icons.

3. JQuery: It's a JavaScript library that provides an easy to use API to make things like HTML document traversal and manipulation, event handling and Ajax much easier and less error prone. Used in the project for all the client side programming.

4. Parsely.js: A client side JavaScript based form validation framework.

5. jsPlumb.js: It's a library that provides API to create visual graphs on the UI by connecting elements on the web page. Used in the project to create visual ontology graph from JSON data produced by server.

6. jquery.form.js: It's a library that can be used for using simple HTML forms as Ajax forms. Used in the project because all the communication between client and server is asynchronous. As shown in the architecture diagram, the entire client functionality is divided into six modules discussed below and Figure 4.2 shows the client work flow diagram.

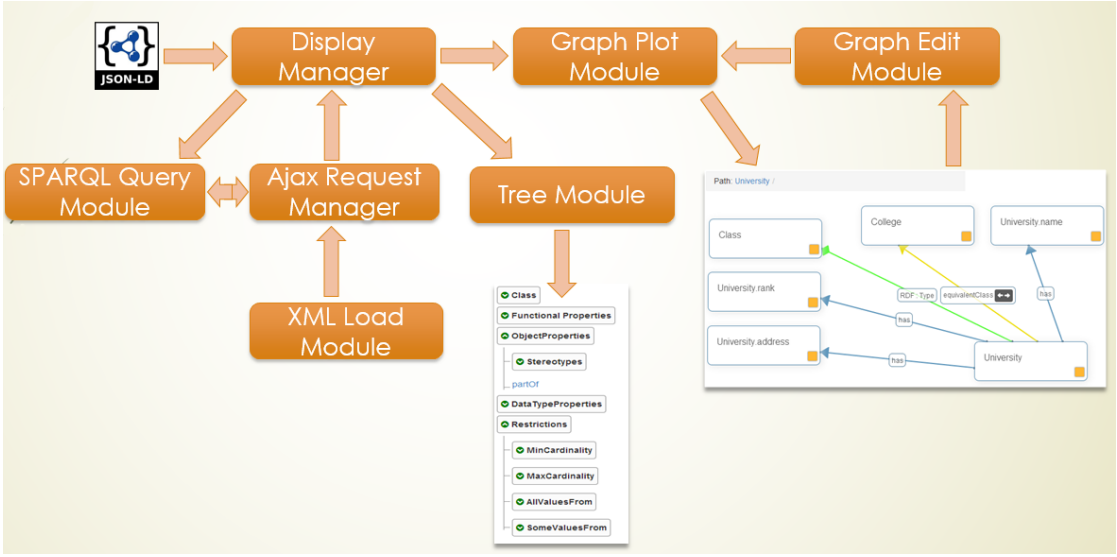

Figure 4.2: client Workflow

As shown in the architecture diagram, the entire client functionality is divided in six modules discussed below.

4.1.1 XML Load Module

This module is implemented using HTML and JavaScript and does two things. First it gives a form to select XML file from the user system and send the file to Ajax request manager to send it to server. The second thing it does is convert the XML file to JSON object which is used by the display manager to display a tree of UML Objects on the page.

4.1.2 SPARQL Query Module

This module is also implemented using HTML and JavaScript and provides a text editor for users to write and submit SPARQL query along with the response format they wants. This data is then sent to Ajax request manager to execute on the server. It then saves the response from the server on the client machine.

4.1.3 Display Manager

This module is completely implemented in JavaScript. Its role is to update the display based on the changes from user or with updated data from server. For example it takes care of updating

the UML tree whenever a XML file is uploaded by the server or update the graph when user edit something on the graph.

4.1.4 Ajax Request Manager

This module takes care of all the communication between client and server. All form requests are processed through this and is implemented only in JavaScript using the jquery form library.

4.1.5 Graph Plot Module

This module is responsible for plotting the ontology graph on page using the response from server. The module implemented in JavaScript using jsplumb library creates the nodes dynamically using the JSON data. Every time a user clicks on a OWL node in the tree the module extracts the node from JSON object finds all its relations creates the nodes and joins them according to the relations between them.

4.1.6 Graph Edit Module

This module completely implemented in JavaScript takes care of updating the client side representation of OWL model and sending the updates to server. When a user updates or adds new mappings using the display they are picked up by this module which first updates the in memory JSON object representing the model then sends the request to Ajax request manager to propagate the updates to server. This module works along with the graph plot module to allow users to edit or add mappings passing control to each other this taking control when data is to be updated and the other one the objects are to be dynamically added on the display.

4.2 Server implementation

Server application is implemented using JAVA programming language and is a web application deployed on Tomcat server. Server application converts the UML model to OWL model, generates mappings between the source models, saves the mappings on the disk and provides a SPARQL query execution environment. All the services provided by the server are in the form of REST endpoints. Table 4.1 outlines the services and their corresponding endpoints.
Following libraries and frameworks were also used for the implementation.

Service	Endpoint
UML to OWL conversion and Mapping Generation	"/generateOWL"
Downloading Mappings	"/mappings"
SPARQL Query service	"/sparqlQuery"
Get default model mappings	"/getDefault"

Table 4.1: System Services and Corresponding endpoints

1. Spring Framework: It provides a configuration model to build MVC and REST applications as required by the project.

2. Maven: It's a software project management and comprehension tool used here for projects dependency management.

3. JENA: It's a Java framework for building Semantic web applications used here to create ontology model in OWL.

4. Drools: It's a business logic integration platform which is used to integrate rules in the system.

As shown in the architecture diagram, the server application is designed in modules. Figure 4.3 shows the interaction diagram for the modules. Figure 4.4 shows the server work flow diagram and implementation of each of the modules is discussed below.

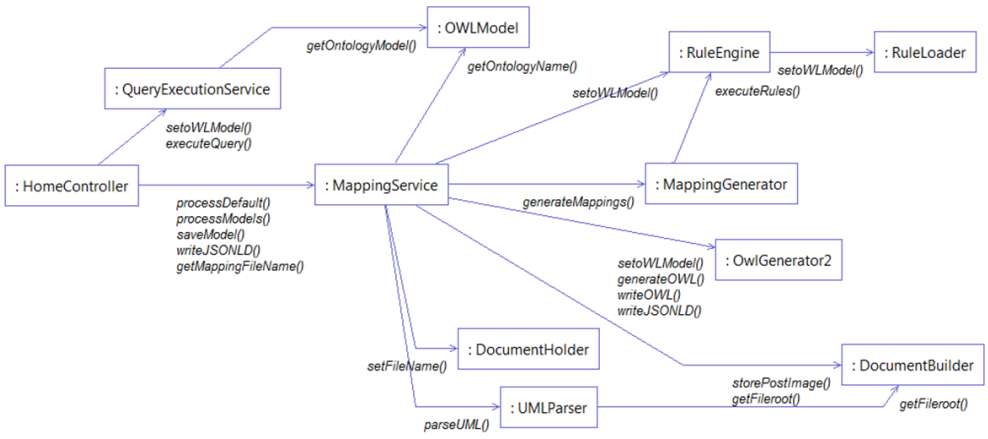

Figure 4.3: Server Modules Interaction Diagram

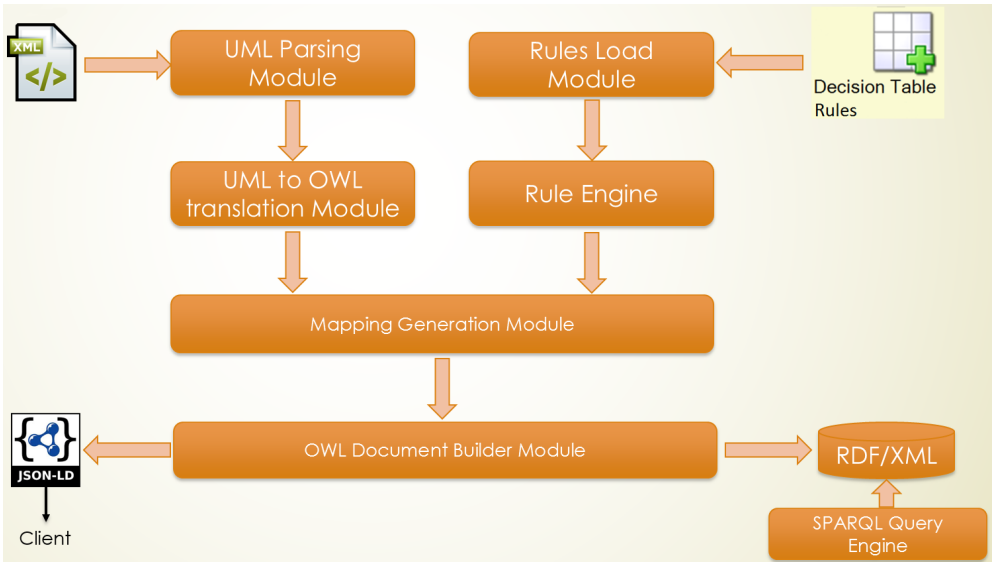

Figure 4.4: Server Workflow

4.2.1 XML and UML Parsing Module

This module parses the UML model in XMI (XML Metadata Interchange) and translates it into a Java object graph. The parsing module takes the UML objects and translates them to internal objects. The translation is done based on the XMI 1.1 DTD, MOF 2 and UML 2 specifications. While converting a UML object translates to a Java object and its attributes to corresponding Java object's fields, creating an in memory graph of objects as the translation progresses. Figure 4.5 shows the class diagram for the parser.

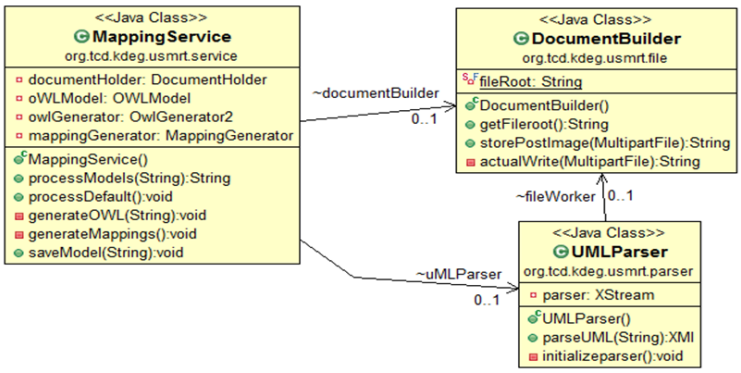

Figure 4.5: UML Parser Class Diagram

Figure 4.6 shows the example of "XMI.header" element and Figure 4.7 shows the corresponding translated objects' class diagram. For the complete class diagram of all the objects with all the

relations please refer appendix B.

Element:	XMI.header
Parents:	XMI |
Children:	XMI: XMI.documentation | XMI.import | XMI.metametamodel | XMI.metamodel | XMI.model |
Comment:	XMI.header contains documentation and identifies the model, metamodel, and metametamodel

Figure 4.6: XMI.header Element According to XMI 1.1 DTD

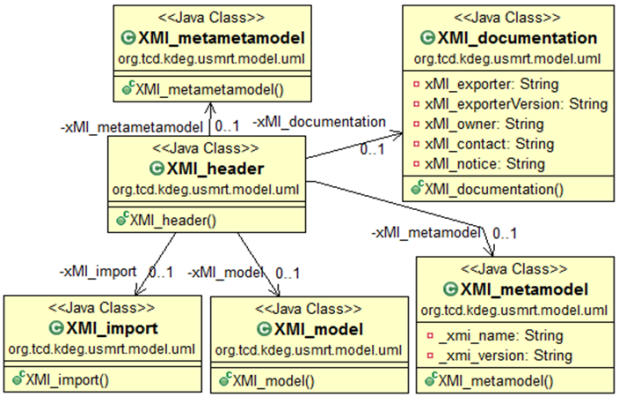

Figure 4.7: Translated XMI.header Class Diagram

4.2.2 UML to OWL Translation and Generation Module

UML to OWL translation is based on the ODM 1.0 specifications. ODM specifications present how concepts from UML match to concepts, in OWL and how the concepts that don't match can be translated. For example a UML class is mapped to OWL class and the attributes of the class are translated to OWL properties with domain as the owner class and attribute type as the range. Attributes with native types get translated to datatype properties while attributes with type of another class are translated to objecttype properties. Table 4.2 shows a summary of features of OWL and UML that can be directly mapped between the two. Building translation on top of standard specifications makes the translation more uniform and has a higher probability of being portable. Figure 4.8 shows the class diagram for the translation Module and Figure 4.9 shows the overview of the class diagram for the internal object graph of OWL model.

Table 4.2: Summary of UML Elements Mapped to OWL Elements

UML elements	OWL elements
class, property, ownedAttribute	class
ownedAttribute, binary association	property
subclass, generalization	Subclass,subproperty
N-ary,association, association class	class,property
Enumeration	oneOf
disjoint, cover	disjointWith,unionOf
Multiplicity	minCardinality,maxCardinality

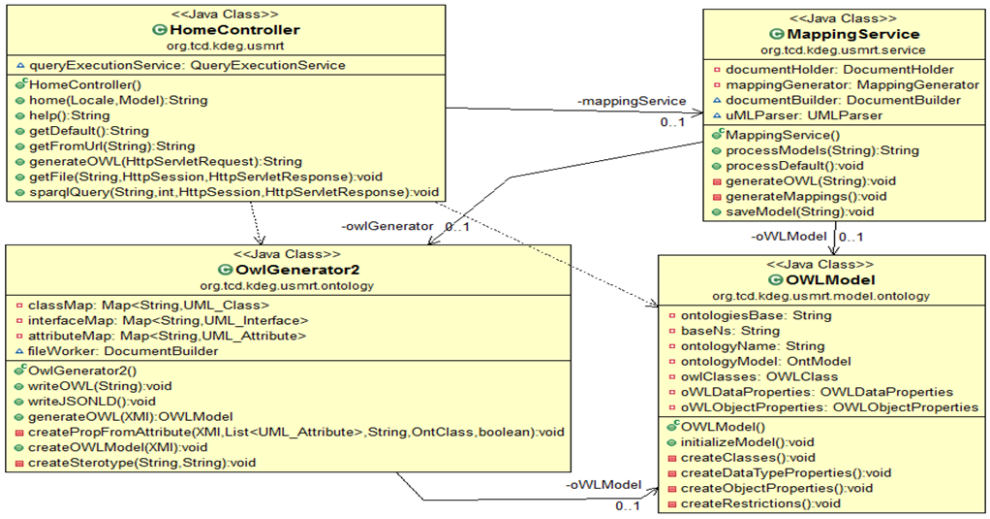

Figure 4.8: UML to OWL Translation Module Class Diagram

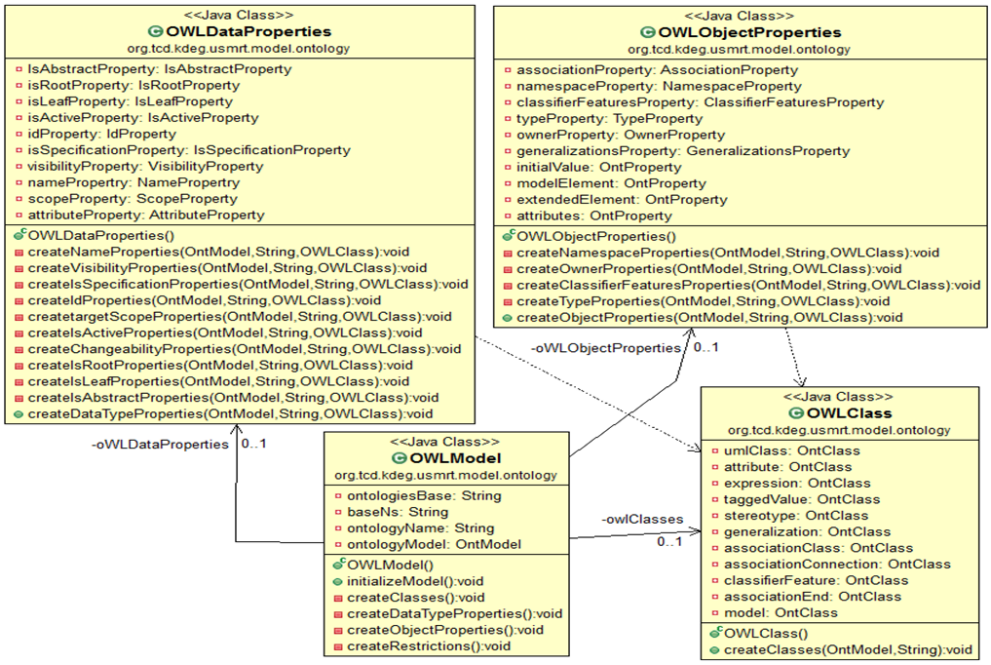

Figure 4.9: Internal OWL Model Overview

JENA Java API is used to generate OWL. All elements from the two XML documents are converted to OWL objects and stored in an in memory ontology graph. This graph is then updated with the generated mappings as discussed in the next two sections. The final graph is then sent to the client in JSON format which is used by the graph plotting Module of the client to create graphs.

4.2.3 Rules Module and Rule Engine

USMRT uses rules to create mappings. The system consists of a set of predefined rules, for example, if two classes are considered equivalent then their subclasses are also equivalent. Apart from the default rules system allows for domain experts to provide rules for mappings using a set of constructs provided by the tool which are then converted to formats used internally by the tool. For example, users can set rules like class “A” is equivalent to class “B”, if both the classes has instances of class “C”. This rule will be loaded by the rules load module and then applied by the rule engine to all the classes at runtime, and the classes which satisfy the rule will be marked equivalent. JBOSS Drools library is used for rule engine which uses RETE algorithm to find the candidates where the mapping rule can be applied matching data tuples against the rules. This strategy allows the tool to use domain specific knowledge to generate more mappings accurately

and allows the tool to be used with application specific rules. Rules are provided by the users are given preference to the default internal rules which means that users can override the default rules if the application requires them to. Users can also provide the rules in the form of an excel decision table. The rules are also allowed to be in a domain specific language with descriptions that can be used to convert it to rules format used by the tool. Figure 4.10 shows the format to be used for specifying rules in a domain specific language. Apart from the specified rules the tool also uses relations inferred by JENA. Figure 4.11 shows the class diagram for the module.

Language Expression	Rule Language Mapping	Object	Scope
There is an Instance with field of "{value}"	i: Instance(field=="{value}")		[condition]
Instance is at least {number} and field is "{value}"	i: Instance(number > {number}, location=="{val...		[condition]
Log : "{message}"	System.out.println("{message}");		[consequence]
Set field of instance to "{value}"	i.setField("{value}");		[consequence]
Create instance : "{value}"	insert(new Instance("{value}"));		[consequence]
There is no current Instance with field : "{value}"	not Instance(field == "{value}")		[condition]
Report error : "{error}"	System.err.println("{error}");		[consequence]
Retract the fact : '{variable}'	retract({variable}); //this would retract bound v...		[consequence]

Figure 4.10: Sample rule statements for rules in domain specific rules.

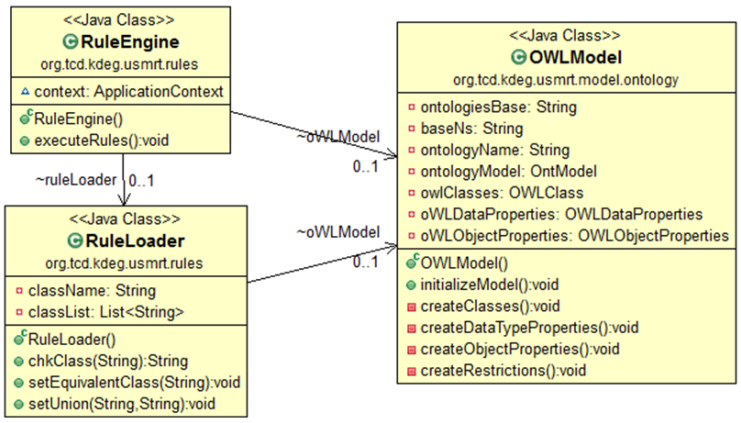

Figure 4.11: Rules Module Class Diagram

The current implementation supports simple, direct mapping and complex direct mapping rules which can be easily extended to support other mapping types. Figure 4.12 shows an example of a simple, direct mapping type. Simple, direct mapping types represent the situations where concepts from one model correspond to another single concept from second model. These include things like equivalent class, subclass etc. Figure 4.12 shows an example of equivalent class.

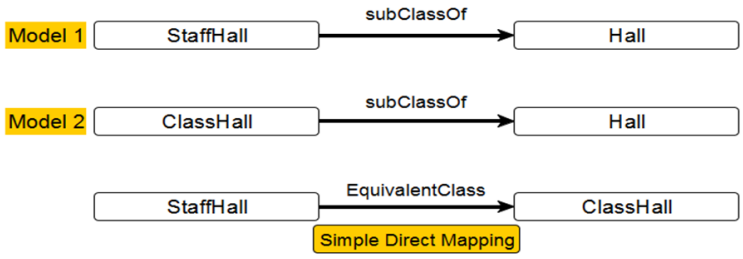

Figure 4.12: Simple Direct Mapping Type

Figure 4.13 shows an example of complex direct mapping type. Complex direct mapping types represent the situations where concepts from one model correspond to multiple other concepts from second model. These include things like unionOf relation, intersectionOf relatation etc. Figure 4.13 shows an example of unionOf relation.

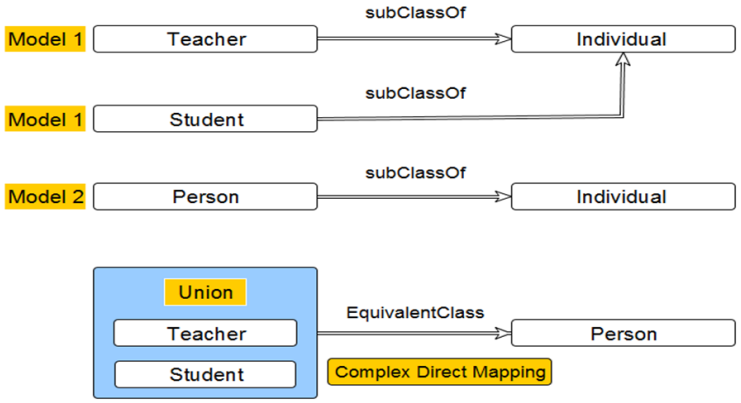

Figure 4.13: Complex Direct Mapping Type

4.2.4 Mapping Generation Module

This module takes the ontology model generated by the OWL generation module and fires the rule engine to execute the rules on the model and then updates the model with the mappings and stores it in file. Figure 4.14 shows the class diagram for mapping generation module.

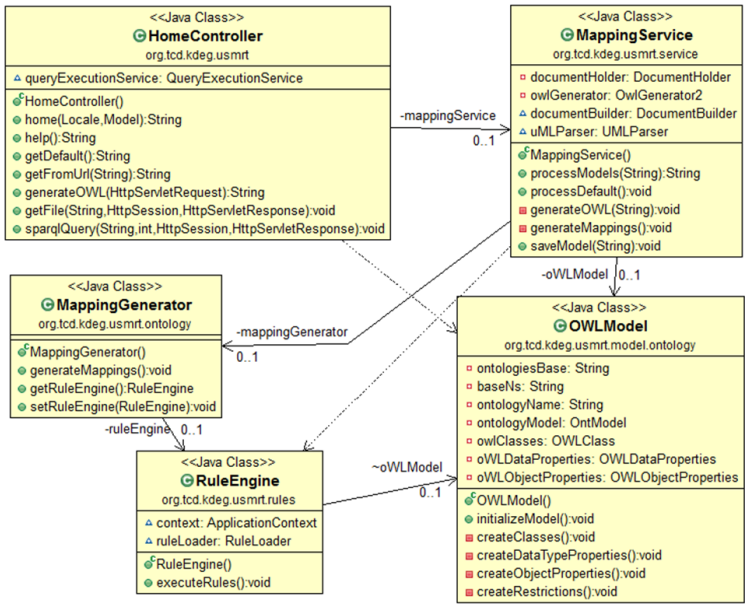

Figure 4.14: Mapping Generation Module Class Diagram

4.2.5 OWL Document Builder

This module saves the ontology model and the mappings in an OWL document in default TURTLE format. It is also responsible to convert the model to JSON format for the client and also responsible for returning the document in downloadable format as requested by the client. Document builder currently supports RDF/XML, TURTLE, JSONLD and RDF/JSON formats and uses JENA API for building format specific documents. Figure 4.15 shows the class diagram for the module.

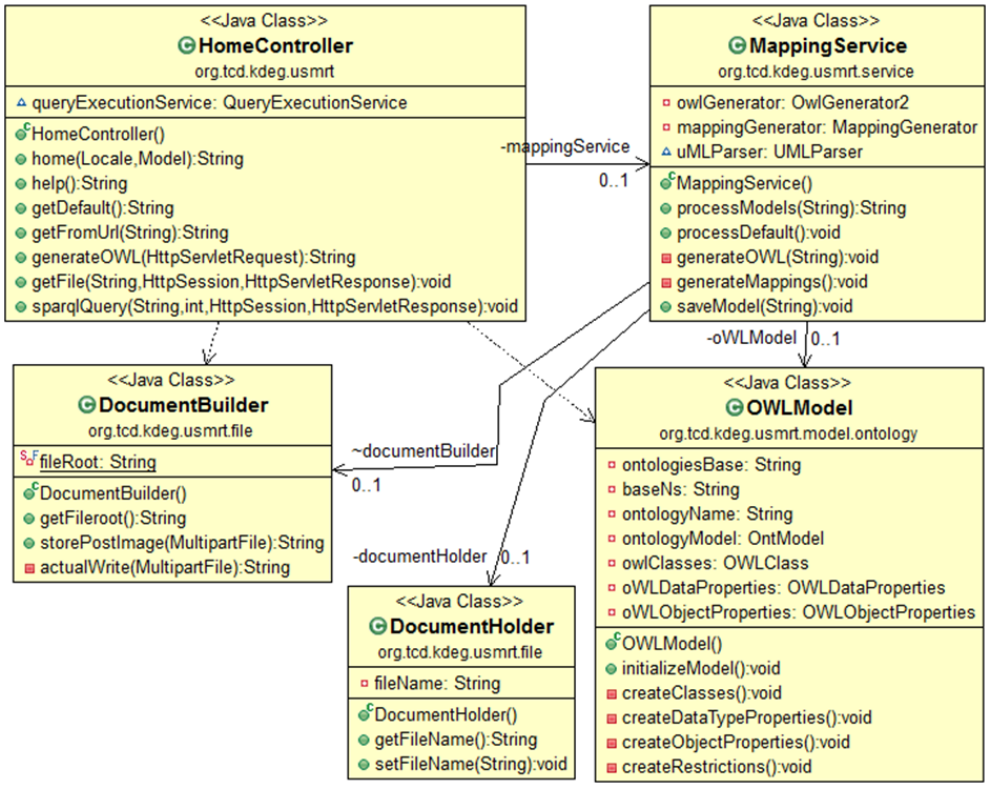

Figure 4.15: OWL Document Builder Class Diagram

4.2.6 SPARQL query Engine

This module provides an execution environment for executing SPARQL queries to query the generated model. It takes the query from the client formats it and executes it on the model. For executing the query it loads the model in memory and uses it for the entire session of a client and returns the results in format requested by user. The module supports TURTLE, N-Triples, RDF/XML, RDF/JSON, XML, JSON, CSV and TSV formats for SPARQL query results. Figure 4.16 shows the SPARQL query engine class diagram. To enable querying JENA API is used.

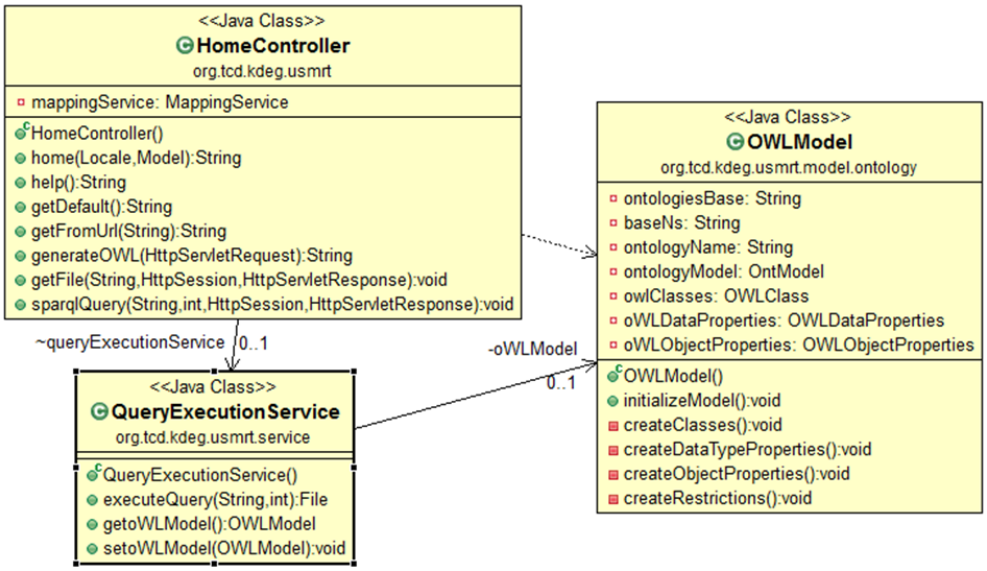

Figure 4.16: SPARQL Query Engine Class Diagram

4.3 Summary

This chapter discussed the implementation details of the USMRT tool including the libraries and languages used, the technical architecture and the details of each of the modules implemented. The discussion of how the modules interact with each other and he workflow were also discussed to give an idea of how the tool works. The next chapter will discuss the evaluation of the tool.

Chapter 5

Evaluation

This chapter presents the experiments performed with the goal of addressing research objective of this dissertation, specifically the evaluation objective. The chapter is structured as follows. First section presents the goals of the experiment. Second section presents the details of the experiment, including the experimental method, participant details, procedure and data collected followed by the experimental results and analysis in the third section and finally the summary in the last section of the chapter.

5.1 Experiment Goals

Several goals were formulated from the questions mentioned above and the experiments were designed to provide data on each of the following goals

1. The tool will correctly convert UML to OWL
2. The tool is able to produce mappings correctly.
3. The tool will make creating mappings easier and faster for users with little expertise in semantic mapping as well as power users.

5.2 Experiment Setup

This section discusses the details of how the experiment was performed. First, subsection gives the details of the participants in the experiment. Second, subsection gives the experimental procedure and the materials collected during the experiment.

5.3 Participants

The experiment required the participants to have some knowledge of UML and/or OWL and/or semantic mapping process. Due to this requirement the experiment was open to postgraduate students, doctoral and post-doctoral researchers of the KDEG group of the computer science and statistics department of the Trinity College Dublin. Participants were recruited by sending a call for participation e-mail to the KDEG group and MSC mobile and ubiquitous computing mailing list. A total of 14 people responded to the emails of which 11 did the evaluation.

The participants who took part in the evaluation were divided into three groups based on their expertise in UML, OWL and mapping development. This division was done to understand how usable the tool was to different target users with different experience level in UML and semantic mappings i.e. To find out if the tool fulfills the research objective to allow users without or less ontology experience (system/software engineers) to create mappings easily and efficiently and at the same time test if it's good for use by the power users with good experience in semantic mapping technologies.

1. Users with experience working with both UML and OWL. This group consists of users who have skills in both UML and OWL (the power users) and thus are considered most important for mappings collected from them that help in understanding the mapping users except for the test models and check if the tool was able to create them correctly and also if the tool is usable by power users.

2. Users with experience working with UML but not OWL. Users in this group are the actual audience of the tool as they do not have experience with Ontology modeling but have UML modeling experience. They represent the software/system engineers who could benefit from the creation of mappings. This group is considered most important for feedback related to the usability of the tool and correctness of the conversion.

3. Users with experience working with OWL but not UML. This group represents the knowledge engineers who experience with ontology mappings techniques and tools and so this group is considered more useful for feedback about mappings representation and usability of the tool.

Users were divided into the above groups based on their answers to questions about their expertise in each field in the survey that followed the experiment. These questions and results are discussed in details in the result and analysis sections. Table 5.1 shows the group division and number of users in each group.

Table 5.1: User Group division based on technical experience

Group Name	Technical Experience	Representing target user type	Number Of Users
Uml only	UML	System engineers	4
Ontology only	OWL, Semantic Mapping	Knowledge Engineers	3
All	UML, OWL, Semantic Mapping	Intersection of the above two	4

5.4 Procedure

Before the experiment ethics approval was taken from the college ethics committee granted on the 2nd of June 2014 and during the same time participants were recruited by emails. Following the ethics approval experiment was carried out over the next two weeks. The participants had the choice of doing the experiment on their own machines using a browser or in the college lab. All the participants chose to do the evaluation on their machines. Before the evaluation each of the participants was contacted by email to fix a time and date of the evaluation and a schedule was prepared to have all the participants perform the experiment at different times without overlapping with each other. The experiment was performed over a period of 11 days from 8th June 2014 to 18th June 2014. After deciding the date and time of the experiment each of the participants were given through email a document describing the experiment, a help document and a URL to a demo video showing the working of the tool. Apart from the above help material the mail also contained the researchers' college email-id, Skype id, Google chat id, and phone number using any of which the participants could contact the researcher for help and questions during the experiment. The email, experiment documentation, help document and the URL for the video is available in Appendix C. The actual video is also available in accompanying DVD. Before the experiment the system was preloaded with two test UML models shown in figure 5.1. In XMI (XML Metadata interchange) format created using open source StarUML tool version 5.0.2.1570.

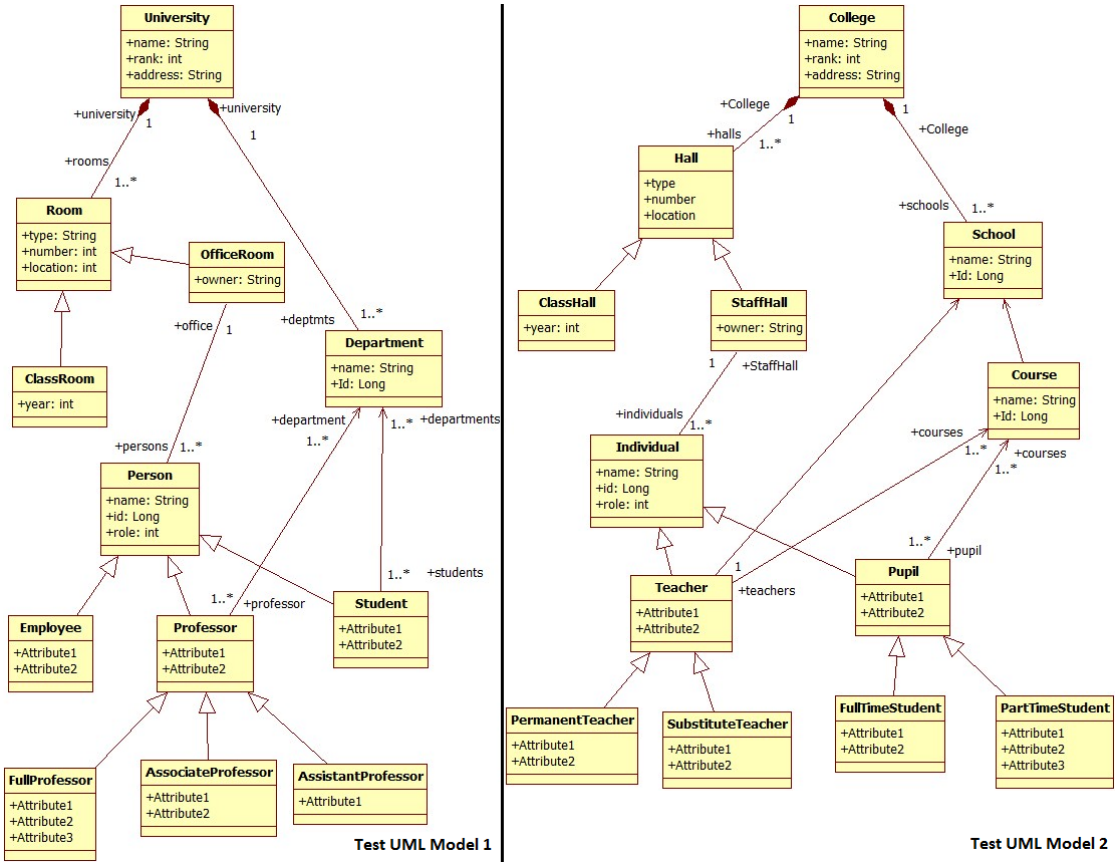

Figure 5.1: Test UML Models

Figure 5.2 shows the steps participants were asked to follow during the course of the experiment. The steps were sent to them through email at the start of the experiment and also explained on instant message (Skype, Google chat) to the users who used it during the experiment.

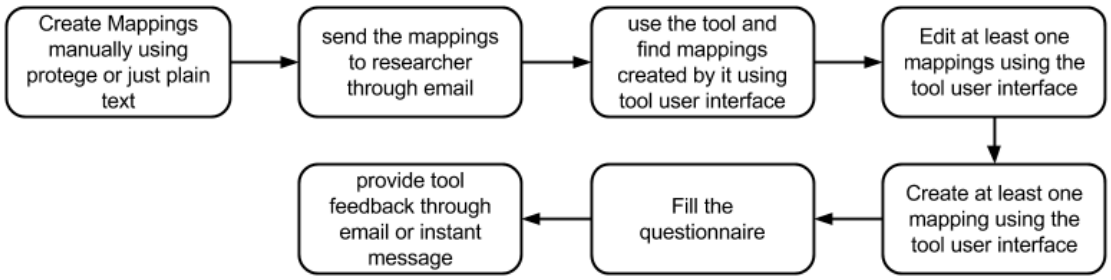

Figure 5.2: Experiment Workflow

Instructions for each of the steps during the experiment were given by email or the instant message (IM) window depending on the conversational medium chosen by the individual participant. Any questions from the participants during the experiment were also received and answered

in similar manner these were saved in file corresponding to the users, are discussed in the analysis section and are also on the DVD. The instruction sheet for steps and participants' questions is provided in Appendix C.

UML models shown in figure 5.1 were also sent to the participant as images in the same email at the beginning of the experiment. The participants were required to create mappings between the two models without using the tool, by either using Protégé or writing them in plain English in owl terminology and send them back to the researcher by replying to the email. These mappings are provided in the result analysis section and the raw mappings in text files are on the accompanying DVD. These mappings were used to understand what kind of mappings, different users sought in the test models. This was used to check how close the tool was to the users' expectations in terms of mappings created by it and was also done to highlight the inherent feature of semantic mapping where different users can create different mappings in the same model and so it's not possible for the tool to create all the mappings that are correct from all the perspective. But at the same time the tool should at least be able to create the ones common to different users. For calculating actual accuracy, the mappings created by tool are compared to the gold standard mappings created by the modeler of the test models (the author), which are also discussed in analysis section and given in Appendix C. Once the mappings were received on email the participants were asked on the conversational medium to use the tool and find the mappings created by it on the user interface. In the same way participants were next asked to perform the tasks of editing some mappings followed by creating some fresh mappings with the tool user interface. Participants' interaction with the tool during both these steps were recorded automatically using JavaScript and was used to calculate the time taken by them on various tasks to understand the parts of the system that were easy to work with and what were not. The logs are available in text format with user numbers in the DVD and the time taken for each task is discussed in the analysis section. As the next step the participants were requested to fill a questionnaire which had questions focusing on evaluating the participants' skill level, their experience with the tool and system usability questions. The link to the questionnaire was provided on the instruction sheet. The questionnaire was used to evaluate usability. The questions are discussed in the results section and the spreadsheet containing the questionnaire with answers and the time taken by users to perform mappings editing and creating task is also available in the accompanying DVD. Finally the participants were asked to provide some feedback about the tool by email or IM according to what they used during the experiment. Suggestions and feedback about the tool were collected to understand what the users liked about the tool and what they thought was missing and could be improved. For this the users were told

about the design and asked about things they thought could be different. All the user feedback is provided in the analysis section. Appendix A contains the table of contents of the accompanying DVD.

5.5 Experiment Results and Analysis

This section presents the results obtained in the experiment and the analysis of the results. All the users that took part in the evaluation answered the questionnaire and by analyzing the results it was found that out of the total 11 users 4 had experience in all UML, OWL and semantic mapping. Another 4 had more experience in UML than in OWL and Semantic Mapping and the final 3 had more experience in OWL and semantic Mapping compared to UML. These results were based on the answers provided by the users about their experience with each of the technologies.

The division into groups was based on following. First the users that use a technology regularly are considered to have higher experience in that technology. Second, if the frequency of use of one technology is more than other than the user is considered to have more experience with that technology for example, user 1 uses OWL, RDF and Semantic mapping Regularly but rarely uses UML so he is put in group “ontology only” while user 2 uses UML sometimes but is only learning OWL and Semantic mapping so he is put in group “email only”. Figure 5.3, 5.4, 5.5 show the response from the users showing their expertise level with each of the technologies.

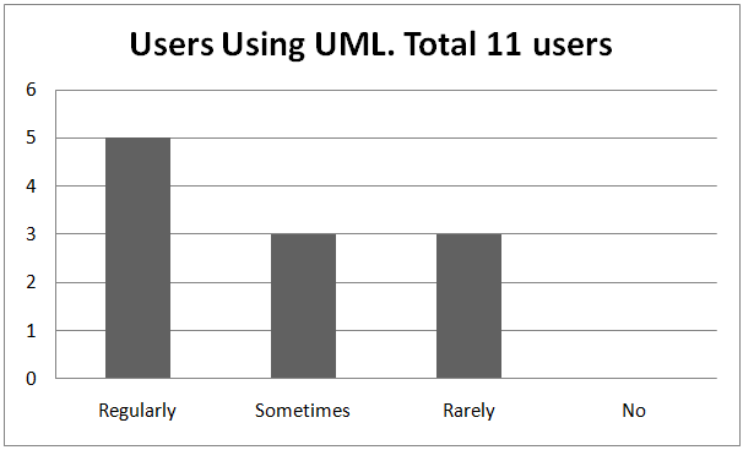

Figure 5.3: UML Experience among Users

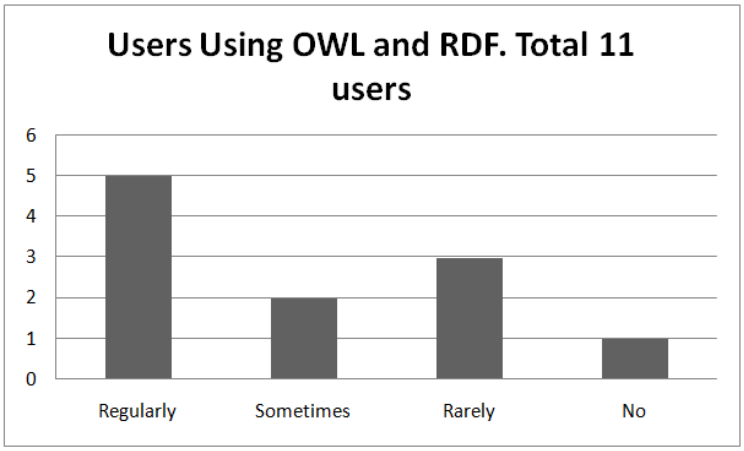

Figure 5.4: OWL and RDF Experience among Users

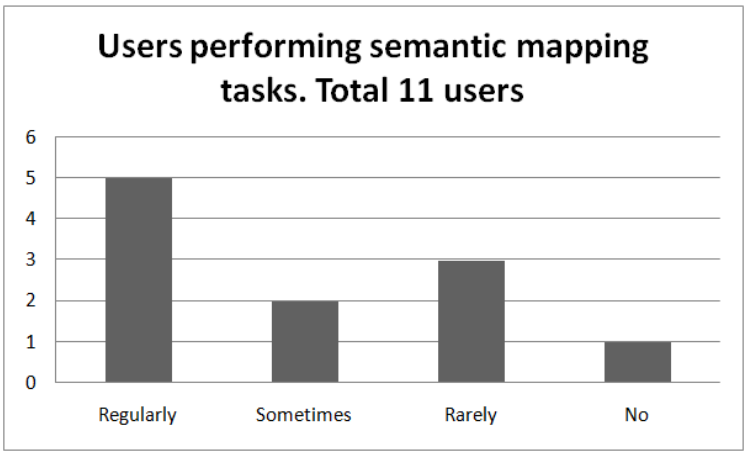

Figure 5.5: Semantic Mapping Experience among Users

The responses to rest of the questions were used to evaluate the overall research goals. Analysis for each of the goal is discussed next.

5.5.1 Accuracy and Effectiveness.

Accuracy and effectiveness is measured in terms the tool's ability to convert UML to OWL and its ability to create mappings between the two models. To check UML to OWL conversion accuracy, users were asked to go through the OWL model using the tool's user interface and tell if the system correctly converted UML to OWL. Figure 5.6 shows user responses to question after looking at the OWL model.

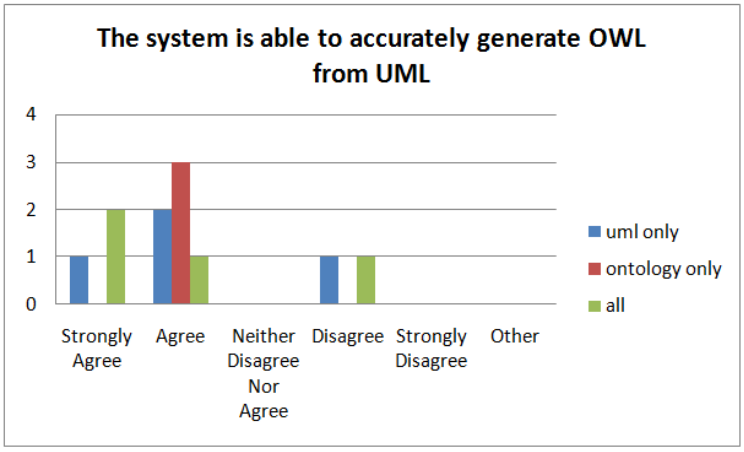

Figure 5.6: Number of users in each group type who said system translated UML to OWL correctly

The responses showed that 9 out of 11 users found created OWL models to be correct while 2 did not. Of the 9 users that agreed that the conversion was correct 3 were from the group "uml only", 3 were from the group "ontology only" and 3 were form group "all". Of the users that said that conversion was incorrect 1 was from the group "uml only" and another one from the group "all". These results showed that the conversion was correct according to 3 out of 4 users representing system engineers, all the users representing knowledge engineers and the 3 out of 4 users representing the intersection of two. To evaluate the mapping creation accuracy mappings created by the tool were compared to the gold standards as shown in table 5.2. Table 5.2 also shows the number of users from each of the user groups who created the same mappings without using the tool. From the table it can be seen that the tool was able to find 8 out of 11 simple direct mappings from the gold standards. Looking at the table it can also be seen that same 8 out of 11 mappings are the most common ones created by users in all the different groups where all or all but 1 user from every group created the mappings rest 3 mappings were created by either 1 or no users and so all 11 mappings are considered for accuracy calculation and the results show that the tool is able to create 72.72% of the simple mappings correctly. From the table, of the 3 complex gold mappings the tool was able to create 1. Only 2 out of 11 users created the same mapping manually and only 1 out 11 created the other two. Since at least one user created all the 3 complex mappings all of them were considered and the results show that tool achieved 33% accuracy for complex mappings. Combining above two the tool created 9 out of total 14 mappings thus an accuracy of 64%. Participants were also asked to go through the mappings created by tool using the user interface and tell if the mappings were correct. Figure 5.7 shows the user response to the question.

Table 5.2: Mapping Comparison Chart

Gold Mappings	Mapping Type	Generated by Tool	"Uml only" group users	"Ontology only" group users	"All" group users
University equivalentClass College	Simple Direct	Yes	4	3	4
Room equivalentClass Hall	Simple Direct	Yes	3	2	4
Person equivalentClass Individual	Simple Direct	Yes	4	3	4
Teacher equivalentClass Professor	Simple Direct	Yes	3	2	2
Student equivalentClass Pupil	Simple Direct	Yes	3	2	4
ClassRoom equivalentClass ClassHall	Simple Direct	Yes	4	2	4
OfficeRoom equivalentClass StaffHall	Simple Direct	Yes	4	2	4
Department equivalentClass School	Simple Direct	Yes	3	2	4
FullProffessor equivalentClass PermanentTeacher	Simple Direct	No	1	0	1
AssociateProfessor equivalentClass PermanentTeacher	Simple Direct	No	1	0	1
AssistantProfessor equivalentClass SubtituteTeacher	Simple Direct	No	1	0	1
Person equivilentClass Pupil union Teacher	Complex Direct	Yes	0	0	2
Pupil equivilentClass FullTimeStudent union PartTimeStudent	Complex Direct	No	0	0	1
Professor equivilentClass FullProffessor union AssociateProfessor union AssistantProfessor	Complex Direct	No	0	0	1

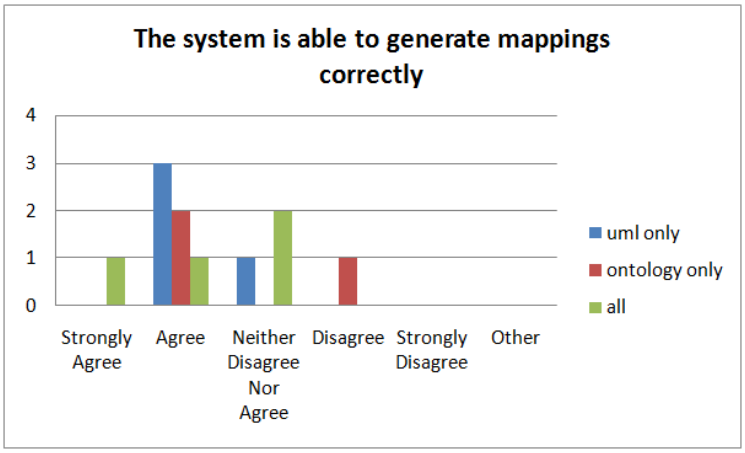

Figure 5.7: Number of users in each group type who said system generated correct mappings

The responses showed that out of the total 11 users 7 said the system correctly generated mappings. 1 user said that the mappings were not correct and 3 others had no opinion. Of the 7 users who said yes 3 were from the group "UML only". 2 were from the "ontology only" group and the other 2 were from "all" group. The user that said no was from the "ontology only" group and of the rest 3 users that had no opinion 1 was from "uml only" group while 2 were from "all" group. The results showed that the system created the mappings correctly according to 3 out of 4 users representing system engineer, 2 out of 3 users representing knowledge engineers and 2 out of 4 users representing the intersection of two.

5.5.2 Efficiency

This includes the time taken by the tool to generate the mappings as well as the time taken by users to edit mappings using tool's user interface and create new mappings with it. In general the tool took less than 5 milliseconds to generate mappings. This time was for the test model which can be considered a small model. In general the time required for tool to generate mappings might depend on the size of the model because of the UML to OWL conversion and for the same reason the time taken by users to edit and create mappings using the tool's interface is also considered for calculating efficiency. Figure 5.8 shows the time taken by all the users to edit the mappings generated by using the tool interface. The mean time was 32 seconds and the mode was 33.

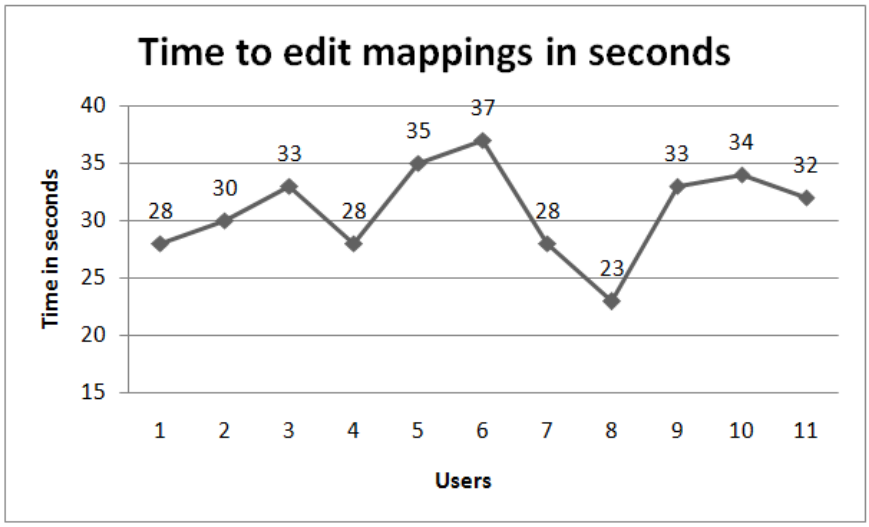

Figure 5.8: Time taken by all users to edit mappings using the Tool's user Interface

Table 5.3 shows the time in seconds taken by users in different user groups to edit mappings. Mean time taken by users from "uml only" group was 32.5 seconds. Mean time taken by users from "ontology only" group was 33 seconds and same for users from "all" group was 31 seconds. From the recorded times it can be seen that tool performed similarly in terms of time taken to edit the mappings by users with different expertise using user interface with a standard deviation 0.849 between the groups. Figure 5.9 shows the time taken by all the users to create mappings using the tool interface. Here the mean was 45.18 seconds and mode was 43. Table 5.4 shows the time in seconds taken by users in different user groups for creating mappings. Mean time taken by users from "uml only" group was 46.25 seconds. Mean time taken by users from "ontology only" group was 45.66 seconds and same for users from "all" group was 43.75 seconds.

Table 5.3: Time taken in seconds by users from different groups to edit mappings using the tool

“uml only” group	“ontology only” group	“All” group
30	28	33
35	34	30
33	37	28
32		33

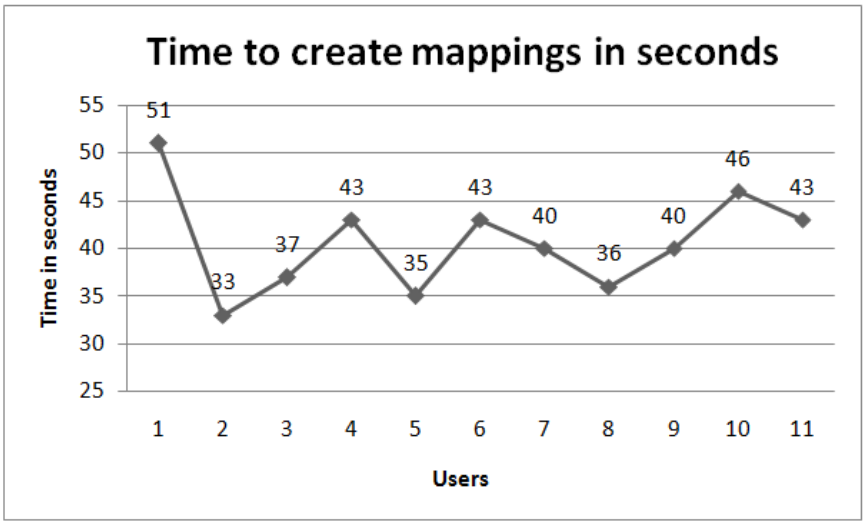

Figure 5.9: Time taken by all users to create mappings using the Tool

Table 5.4: Time taken in seconds by users from different groups to edit mappings using the tool

“uml only” group	“ontology only” group	“All”group
43	41	43
47	40	39
52	46	46
43		47

From the recorded times it can be seen that tool performed similarly in terms of time taken to create mappings by users with different expertise using the user interface with a standard deviation 1.306 between the groups.

5.5.3 Usability

This section evaluates ease of use and other usability aspects of the tool. To evaluate usability users were asked a number of questions and the response was recorded. User responses were used to calculate the System Usability Scale (SUS) score and a score of 70.45 was received. Among other aspects some of the questions were designed to know if it was easy to relate between UML and OWL models for which 8 out of 11 users said yes. Of these 8 users 3 users were from "UML only" and "All" groups and 2 were from the "ontology only" group. Other question asked if edit mappings were easy for which 9 out of 11 users said yes of which 3 users from each of the groups. Similarly 10 out of 11 users agreed that adding new mappings was easy from the interface. Figure 5.10 shows the summary of user response to questions considered most important for usability. It shows the overall number of users and number of users from each group agreeing positively to the questions. Table of all questions and the user response for each are available in Appendix C.

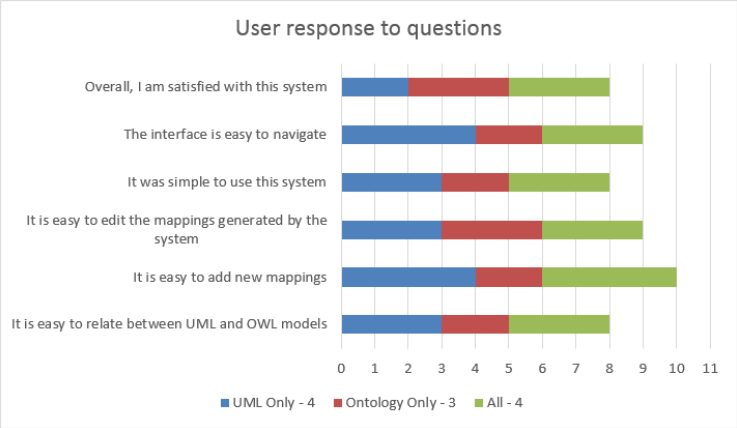

Figure 5.10: Summary of user response to questions.

Apart from the answers to above questions users were also asked for feedback on the tool and suggestions on improvement, the summary of which is provided in table 5.5

Table 5.5: Summery of user feedback, suggestions and questions on tool

Feedback	Suggestions	User	User Group
very user friendly, general hierarchy of the whole thing could be created by anyone with little to no knowledge of data bases I think	Display “hasSubclass”, relation along with “subclassof” relation	User1	Uml only
Overall I really liked the tool. It was easy to navigate and didn't require much training. Clicking on nodes to view subclasses/relationship mappings was very efficient., Nice interface, clean and not too jumbled	Might be nice to use some kind of visualization to distinguish primary node from other nodes, Also, an algorithm to prevent nodes appearing over each other or other links. Finally, navigational aids might be nice, simple zoom in/out for example	User11	Uml only
+Easy to use,+Nice UI,-I sometimes accidentally clicked a node when creating a relationship and the view changed,-Not sure how mapping rules in relationship editor works,+Drag and drop very handy		User10	Ontology only
I think the tool looks great	One thing that I think could definitely improve the tool is some kind of autocomplete list of existing classes.,You know, when you add a new node to a certain view. I wasn't sure if the node I was going to add would refer to the existing node	User6	Ontology only
Browsing through model is very easy	Ability to edit source model would be good	User8	all

5.6 Summary

This chapter discussed the results obtained in the experiments to evaluate different research objectives. The results were evaluated to find the accuracy of the tool in generating mappings and it was found that the tool generated mappings with an accuracy of 64%. Second the efficiency was evaluated to see how much time it tool for the tool to generate mappings and it was found that the tool took less than 5 milliseconds to generate mappings from test model. The time taken by users to edit and create mappings using the tool’s user interface were also evaluated and it was found that the tool performed for all users and the user experience with UML or semantic mapping did not influence to a large extent their ability to edit or create mapping with the tool’s interface. Finally, the usability of the user interface with respect to different target users of the tool was calculated and it was found that the tool performed similarly irrespective of the user experience in UML or OWL.

Chapter 6

Conclusion and Future Work

This chapter discusses the extent to which the objectives of this research were achieved, the contributions made, the future of this research and some final remarks.

6.1 Research Objectives

6.1.1 Develop an approach to extract knowledge models from UML class diagram.

To achieve this objective an approach using Java, Jena, OWL-DL, ODM 1.0 specifications and XMI schema was created and the results were evaluated. From the results see section 5.2 it was found that the approach was able to correctly extract knowledge model expressed in OWL DL convert from UML. The results back up the theory that the approach presented can be used to extract knowledge models from UML class diagrams.

6.1.2 The system can correctly generate mappings between UML models.

This objective was achieved by implementing a rule based approach of mapping creation and the results were evaluated to find the accuracy see section 5.2. From the results in can be seen that the system created 64% of the mappings correctly. The results show that the presented approach works, but more work is required to make it more efficient.

6.1.3 The system is easy to use by both UML users and ontology users.

To evaluate this objective users were divided in categories of UML users, ontology users and users of both and asked to perform tasks with the tools users interface and it was found that the time taken by users from all the groups to edit mappings had a standard deviation of 0.8 and 1.3 for creating mappings which shows that the users experience with a particular technology was not a major factor for using the tool. The results see section 5.3 also showed that more majority of users from each group said that the tool was easy to use. From the feedback received from users see table 5.9 it can be seen that though the tool archives the research objective some improvements are required to make it more usable.

6.2 Contributions

6.2.1 Demonstrates the possibility of using UML for creating semantic mappings between software systems.

Most current research in the field of using UML for knowledge engineering are limited to either using UML for modeling itself or extracting the knowledge from them. Using UML to generate mappings between the models will help in integrating the systems semantically.

6.2.2 Exhibits UML to OWL conversion based on ODM 1.0 specifications and XMI 1.1 DTD.

Unlike most current approaches that use custom logic and changes to either of the technologies to achieve translation. The presented approach will give a more standard and portable translation between the two because its build on standard specifications and standards which will also make it easier to evolve as the standards evolve.

6.2.3 Indicates the possibility of using business rules in automatic mapping generation.

Using rules for creating mappings is a widely used approach. Here the approach is extended to involve business rules. Business rules are the standard way of achieving some behavior in systems which can be changed to get different results without the requirement to change anything in the code. This approach can also be applied to mapping creation where different mappings can be

achieved by simply changing the rules which could make systems infer each other differently at runtime.

6.2.4 Demonstrates a user interface for mapping creation system for use by system and knowledge engineers.

Most mapping tools that exist do not provide a user interface and depend on other interfaces for user validation and improvements of mappings. The interface provided in this work shows how an integrated environment can help users to work more easily and efficiently. It also shows how the mappings and knowledge model can be displayed which is similar to the graphical representation of UML making it easier to relate between the two.

6.3 Future Work

This work was limited to two types of mappings namely simple direct mappings and complex direct mapping and was able to generate these mappings to a certain extent. Next steps in the work are. First, tune the system for better performance on currently supported mapping types and include variations of these mapping types and other mapping types. Second, develop a suitable user interface to allow user to provide mapping rules from the web interface itself and use the user feedback from evaluation to improve the user interface and overall system usability.

6.4 Final Remarks

Involving semantics in software models aims to make the integration processes easier and faster by enabling integration at a higher knowledge level. But, the differences in the semantic models produced by different people present a hurdle in this vision. Semantic mapping aims to solve this, but since it's a specialized field and is not widely used yet in the enterprise, the promises it provide are not being utilized to their full potential. This dissertation presents a system that can help research trying bring semantic mapping to wider use in enterprise by not only using familiar technologies for creating mappings like the use of UML and processes like using decision table for specifying business rules that govern integration and are standards and commonly used in industry, but also provide a user interface that represents mappings in familiar and easy to use manner that can be used by domain experts and system engineers with reduced learning efforts. Such an approach, which reduces the learning efforts and simplifies the process for the domain experts

who possess the right expertise for the job can bridge the gap between knowledge engineering and system engineering.

Bibliography

[Choi2006] Namyoun Choi, Yeol Song, and Hyoil Han. 2006. A survey on ontology mapping. ACM SIGMOD Record, vol 35, no 3, pp 34--41, ACM, September 2006.

[Madhavan2002] Jayant Madhavan, Philip A. Bernstein, Pedro Domingos, and Alon Y. Halevy. 2002. Representing and reasoning about mappings between domain models. In Proceedings of the 18th national conference on Artificial intelligence (AAAI 2002). Edmonton, Alberta, Canada, pp 80--86, July 2002.

[Doan2002] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. 2002. Learning to map between ontologies on the semantic web. In Proceedings of the 11th international conference on World Wide Web (WWW 2002). Honolulu, Hawaii, USA, pp 662--673. May, 2002.

[Staab2004] Ehrig, Marc, and Steffen Staab. 2004. QOM–quick ontology mapping. In Proceedings of the 3rd International Semantic Web Conference (ISWC 2004). Hiroshima, Japan, pp 683--697, November, 2004.

[Noy2000] Noy, Natalya Fridman and Musen, Mark A. "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment". In Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence. Austin, Texas, USA, pp 450--455, August 2000.

[Noy2001] Noy, Natalya F., and Mark A. Musen. "Anchor-PROMPT: Using non-local context for semantic matching". In Proceedings of the workshop on ontologies and information sharing at the international joint conference on artificial intelligence (IJCAI 2001). Seattle, Washington, USA, pp. 63-70, August 2001.

[Noy2004] Natalya F. Noy. 2004. Semantic integration: a survey of ontology based approaches. ACM SIGMOD Record, vol 33, no 4, pp 65--70, ACM, December 2004.

[Pinto2001] Helena Sofia Pinto and João P Martins 2001. A methodology for ontology integration. In Proceedings of the 1st international conference on Knowledge capture (KCAP 2001). Victoria, British Columbia, Canada. pp 131--138, October 2001.

[Ngo2013] Ngo DuyHoa, Zohra Bellahsene, and Konstantin Todorov. 2013. Opening the Black Box of Ontology Matching. In Proceedings of the 10th International European Semantic Web Conference (ESWC 2012). Montpellier, France. pp 16--30. May 2013.

[Doan2000] AnHai Doan, Pedro Domingos, Alon Levy. 2000. Learning Source Descriptions for Data Integration. In Proceedings of the 3rd International Workshop on the Web and Databases (WebDB 2000), Dallas, Texas, USA, pp. 81--86. May 2000.

[Mitra2005] Mitra, Prasenjit, Natasha F. Noy, and Anuj Rattan Jaiswal. 2005. OMEN: A probabilistic ontology mapping tool. In Proceedings of the 4th International Semantic Web Conference (ISWC 2004). Galway, Ireland. pp. 537--547. November 2005.

[Ehrig2004] Ehrig, Marc, and York Sure. 2004. "Ontology mapping–an integrated approach. In Proceedings of the 1st European Semantic Web Symposium (ESWS 2004). Crete, Greece. pp. 76--91. May 2004.

[Motik2009] Motik, Boris, Bernardo Cuenca Grau, Ian Horrocks, and Ulrike Sattler. 2009. Representing ontologies using description logics, description graphs, and rules. Artificial Intelligence, vol 173, no 14, pp 1275--1309, Artif. Intell., February 2009.

[Bouquet2003] Bouquet, Paolo, Fausto Giunchiglia, Frank Van Harmelen, Luciano Serafini, and Heiner Stuckenschmidt, 2003. Cowl: Contextualizing ontologies. In Proceedings of the 2nd International Semantic Web Conference (ISWC 2003), Sanibel Island, Florida, USA. pp. 164--179. October 2003.

[Baclawski2001] Baclawski, Kenneth and Kokar, Mieczyslaw M. and Kogut, Paul A. and Hart, Lewis and Smith, Jeffrey E. and Holmes,III, William S. and Letkowski, Jerzy and Aronson,

Michael L. 2001. Extending UML to Support Ontology Engineering for the Semantic Web. In Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools (2001). Toronto, Canada, pp 342--360, October 2001.

[Chan2001] Wang, Xin, and Christine W. Chan. 2001. Ontology Modeling Using UML. In Proceedings of the 7th International Conference on Object Oriented Information Systems, (OOIS 2001). Calgary, Canada, pp 59--68, August 2001.

[We2003] WE1 LIU, ZONGTIAN LIU, KUN SHAO 2003. UML based domain ontology modeling for multi agent system. In Proceedings of the 2nd International IEEE Conference on Machine Learning and Cybernetics (ICMLC 2003). Xian, Shaanxi, China. pp 407--412. November 2003.

[Fu2009] Fu Zhang, Z. M. Ma, Jingwei Cheng, and Xiangfu Meng. 2009. Fuzzy semantic web ontology learning from fuzzy UML model. In Proceedings of the 18th ACM conference on Information and knowledge management (CIKM 2009). Hong Kong, China. pp 1007--1016, November 2009.

[Gašević2007] Milanović, Milan, Dragan Gašević, Adrian Giurca, Gerd Wagner, and Vladan Devedžić. 2007. Sharing ocl constraints by using web rules. Electronic Communications of the European Association of Software Science and Technology, vol 9, no. 11, pp 2--18, EASST, November 2007.

[Staab2010] Parreiras, Fernando Silva, and Steffen Staab. 2010. Using ontologies with uml class based modeling: The twouse approach. Data And Knowledge Engineering, vol 69, no. 11, pp 1194--1207, DKE January 2010.

[Zhuoming2012] Zhuoming Xu, Yuyan Ni, Wenjie He, Lili Lin, Qin Yan. 2012. Automatic extraction of OWL ontologies from UML class diagrams: a semantics preserving approach. World Wide Web, Volume 15, Issue 56, pp 517--545, September 2012.

[Gasevic2004] Dragan Gasevic and Djuric, Devedzic Valdan, and violeta Damjanovic. 2004. From UML to ready to use OWL ontologies. In Proceedings of 2nd International IEEE Conference on Intelligent Systems (2004). Toronto, Canada, pp 485--490, June 2004.

[Kalfoglou2003] Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. The Knowledge Engineering Review Journal (KER), vol 18, no. 1, pp 1--31, ACM, January 2003.

[Shvaiko2005] Shvaiko, Pavel, and Jérôme Euzenat. A survey of schema-based matching approaches. The Journal on Data Semantics IV, vol 3730, pp 146--171, Springer Berlin Heidelberg, 2005.

[Falconer2007] Falconer, S., Storey, M. A. A cognitive support framework for ontology mapping. In Processing of the 6th International Semantic Web Conference (ISWC). Busan, Korea, pp 114--127, November 2007.

[Bernstein2008] Bernstein, Philip A. and Haas, Laura M. Information Integration in the Enterprise. Communications of the ACM. vol 51, no. 9, pp 72--79, ACM, September 2008.

[Dou2006]Dou, Dejing and LePendu, Paea. Ontology-based Integration for Relational Databases. In Processing of the 2006 ACM Symposium on Applied Computing. Dijon, France, pp 461--466, April 2006.

[UML2.0] UML 2.0. Object Management Group (OMG), "Unified Modeling Language: Superstructure version 2.0", `http://www.omg.org/spec/UML/2.0/`, Released July 2010. Retrieved August. 2014.

[UML2011] UML 2011. UML® Resource Page, "Unified Modeling Language", `http://www.uml.org`, Retrieved August. 2014.

[SPARQL2008]SPARQL 2008. W3C (World Wide Web Consortium), "SPARQL Query Language for RDF", W3C Recommendation, `http://www.w3.org/TR/rdf-sparql-query/`, 15 Jan. 2008. Retrieved August. 2014.

[DOM] DOM Core. W3C (World Wide Web Consortium), "Document Object Model Core", W3C Recommendation, `http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113/core.html`, 13 Nov. 2000. Retrieved August. 2014.

[ODM1.0]ODM 1.0. Object Management Group (OMG), "Documents associated with Ontology Definition Metamodel (ODM) Version 1.0", `http://www.omg.org/spec/ODM/1.0/`, Released May. 2009. Retrieved August. 2014.

[OMG2010]OMG 2010. Object Management Group (OMG), "Who's OMG", `http://www.omg.org/marketing/about-omg.htm`, Dec. 2010. Retrieved August. 2014.

[OMG2011]OMG 2011 Object Management Group (OMG), "Catalog of UML Profile Specifications", `http://www.omg.org/technology/documents/profile_catalog.htm`, Oct. 2011. Retrieved August. 2014.

[OWL2014]OWL Overview. World Wide Web Consortium (W3C), "OWL Web Ontology Language Overview", W3C Recommendation, `http://www.w3.org/TR/owl-features/`, 10 February 2004. Retrieved August. 2014.

[RDF] RDF Schema. World Wide Web Consortium (W3C), "RDF Vocabulary Description Language 1.0: RDF Schema", W3C Recommendation, `http://www.w3.org/TR/rdf-schema/`, 10 February 2004. Retrieved August. 2014.

[XMI1.1] XMI Reference. `http://zvon.org/xxl/XMI/Output/`. Retrieved August. 2014.

[JENA] https://jena.apache.org/. Retrieved August. 2014.

Appendix A

Table of Contents of the accompanying DVD

1. Application code.
2. Evaluation Questionnaire Spread Sheet with user Responses.
3. User Mapping Files and Feedback.
4. USMRT Demo video.
5. Participants time log while performing tasks with User Interface.

Appendix B

USMRT Class Diagrams

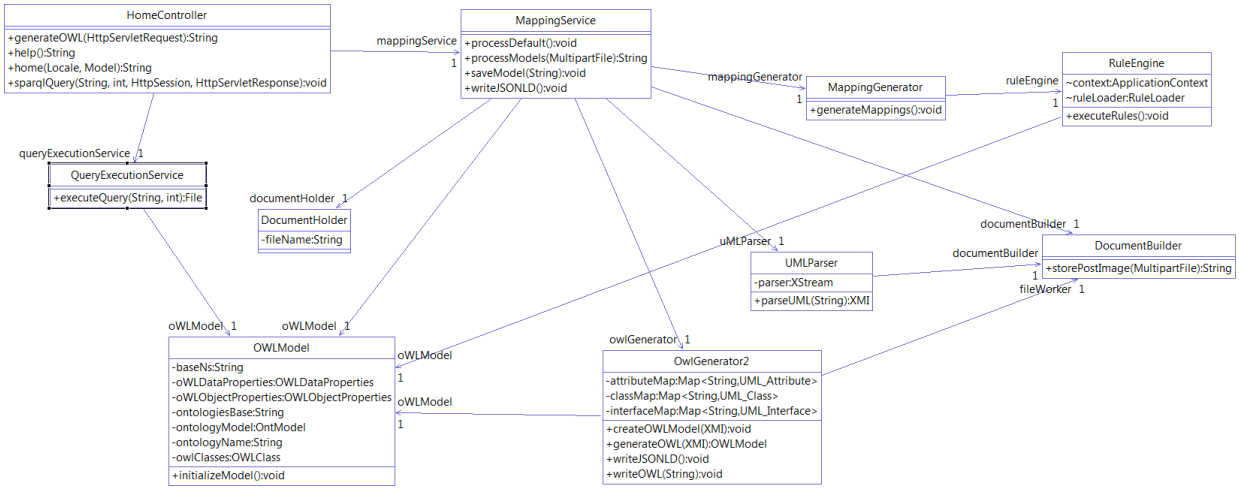

Figure B.1: USMRT class diagram

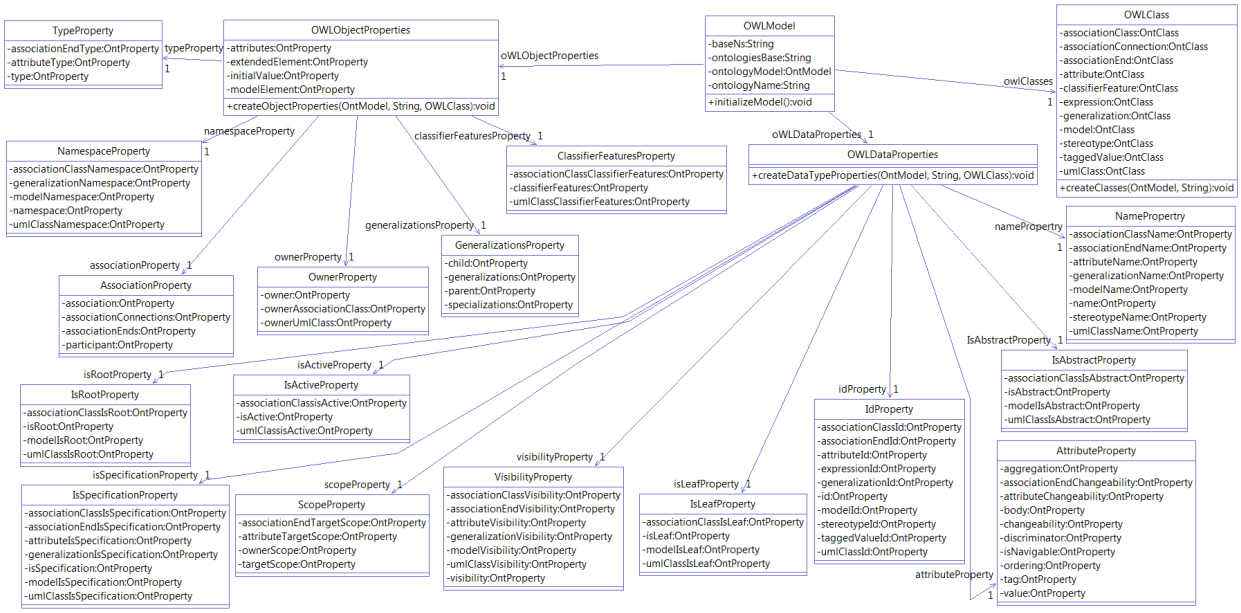

Figure B.2: Internal OWL Model class diagram

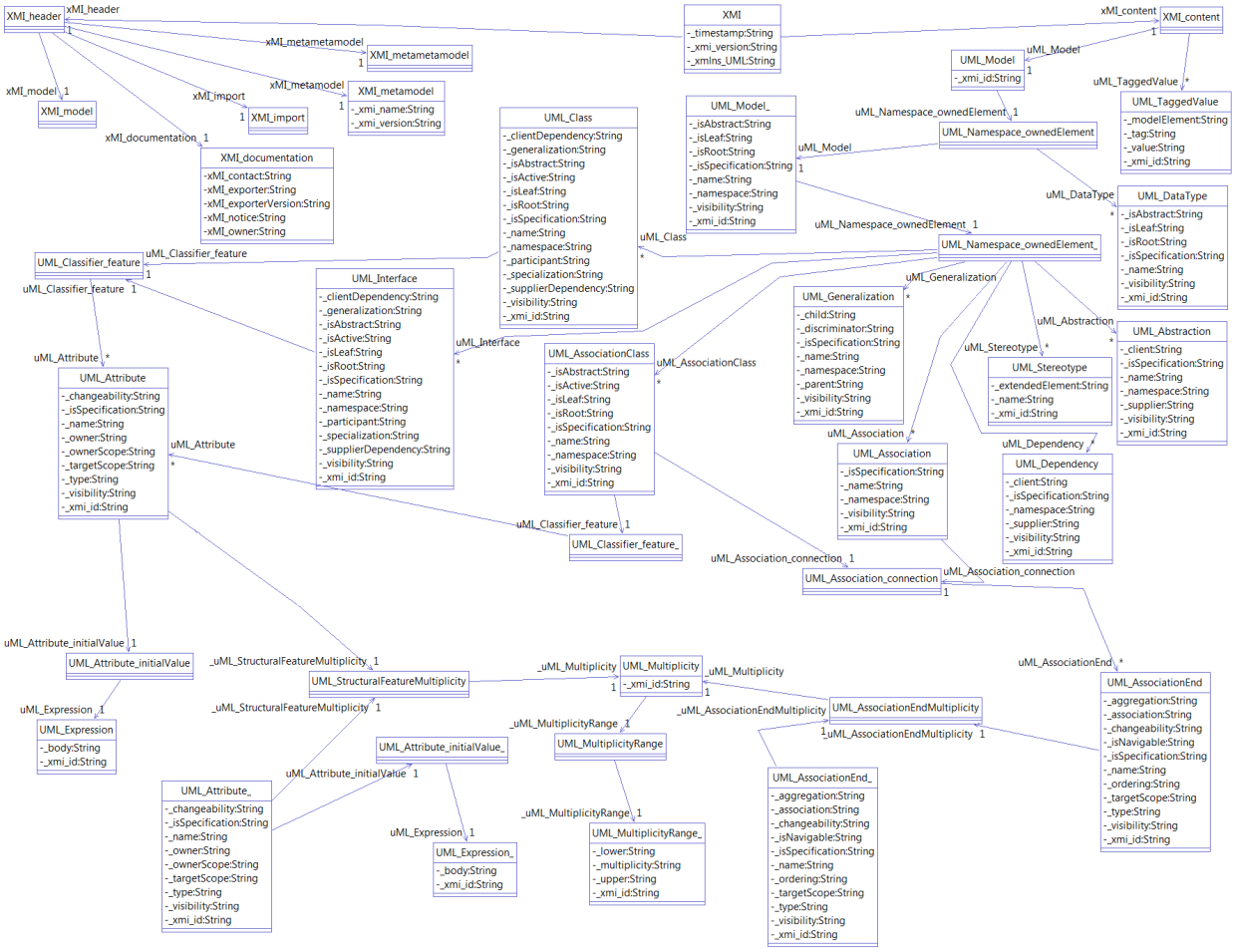

Figure B.3: Internal UML Model class diagram

Appendix C

Evaluation Material

1. Table of usability Questionnaire and user responses.

Table C.1: Overall user response to usability questionnaire

Questions	Strongly Agree	Agree	Neither Disagree Nor Agree	Disagree	Strongly Disagree	Other
It is easy to relate between UML and OWL models		8	1	1		1
It is easy to add new mappings	3	7	1			
It is easy to edit mappings generated by system	4	5	2			
It was simple to use this system	2	6	2			1
It was easy to learn to use this system	1	8	2			
The interface is easy to navigate	2	7	1	1		
The interface of this system is pleasant	4	5	1	1		
I am able to efficiently complete my work using this system	2	2	5	1		1
I can effectively complete my work using this system		4	6			1
Overall, I am satisfied with how easy it is to use this system	1	8	1			1
Overall, I am satisfied with this system		8	2	1		

Table C.2: Response to usability questionnaire from users in group “uml only”

Questions	Strongly Agree	Agree	Neither Disagree Nor Agree	Disagree	Strongly Disagree	Other
It is easy to relate between UML and OWL models		2	1			1
It is easy to add new mappings	1	3				
It is easy to edit mappings generated by system		3	1			
It was simple to use this system		3	1			
It was easy to learn to use this system		3	1			
The interface is easy to navigate	1	3				
The interface of this system is pleasant	1	3				
I am able to efficiently complete my work using this system	1		1	1		1
I can effectively complete my work using this system		1	2			1
Overall, I am satisfied with how easy it is to use this system		4				
Overall, I am satisfied with this system		2	2			

Table C.3: Response to usability questionnaire from users in group “ontology only”

Questions	Strongly Agree	Agree	Neither Disagree Nor Agree	Disagree	Strongly Disagree	Other
It is easy to relate between UML and OWL models		3				
It is easy to add new mappings	1	1	1			
It is easy to edit mappings generated by system	1	2				
It was simple to use this system	1	1	1			
It was easy to learn to use this system		3				
The interface is easy to navigate	1	1	1			
The interface of this system is pleasant	1	2				
I am able to efficiently complete my work using this system			3			
I can effectively complete my work using this system			3			
Overall, I am satisfied with how easy it is to use this system		2	1			
Overall, I am satisfied with this system		3				

Table C.4: Response to usability questionnaire from users in group “all”

Questions	Strongly Agree	Agree	Neither Disagree Nor Agree	Disagree	Strongly Disagree	Other
It is easy to relate between UML and OWL models		3		1		
It is easy to add new mappings	1	3				
It is easy to edit mappings generated by system	2		1			
It was simple to use this system	1	2				1
It was easy to learn to use this system	1	2	1			
The interface is easy to navigate		3		1		
The interface of this system is pleasant	2		1	1		
I am able to efficiently complete my work using this system	1	2				1
I can effectively complete my work using this system		3	1			
Overall, I am satisfied with how easy it is to use this system	1	2				1
Overall, I am satisfied with this system		3		1		

2. Test Model Gold Mappings.

Table C.5: Gold Mappings Between Test Models

Gold Mappings	Mapping Type
University equivalentClass College	Simple
Room equivalentClass Hall	Simple
Person equivalentClass Individual	Simple
Teacher equivalentClass,Professor	Simple
Student equivalentClass Pupil	Simple
ClassRoom equivalentClass ClassHall	Simple
OfficeRoom equivalentClass StaffHall	Simple
Department equivalentClass School	Simple
FullProffessor equivalentClass PermanentTeacher	Simple
AssociateProfessor equivalentClass PermanentTeacher	Simple
AssistantProfessor equivalentClass,SubtituteTeacher	Simple
Person equivilentClass Pupil union Teacher	Complex
Pupil equivilentClass FullTimeStudent union PartTimeStudent	Complex
Professor equivilentClass FullProffessor union AssociateProfessor union AssistantProfessor	Complex

3. Participants questions and doubts during the evaluation.

   (a) user2

   Could you clarify for me what you mean by "mapping" for the class diagrams?
   Do you want me to edit the diagrams in the tool to show the mapping? e.g. department and school classes are not shown as equivalents in the diagrams.

   (b) user11

   should I write down properties and cardinality as well?
   if I access the class University, it had a College node with a relationship that says they're equivalent If I access College, however, I can't see the equivalence I pressed the button Add Node, typed in University and made them equivalent for some reason, every time I access College again, that new node disappears but if I access University, there will be two equivalence relationships between University and College. Is this correct?
   if I try to delete one of the equivalences, it will let me do it, but if I access any other class and then go back, the double equivalence will still be there. It will reappear from nowhere.

If I access the class Teacher it will show me all of its properties and also the Individual node because Teacher is a subclass of Individual but Teacher has subclasses of its own is there a way to visualize them? cuz if I didn't know that previously, I would have to guess.

4. USMRT demo video URL.

   `https://drive.google.com/file/d/0B7qWHaf9smNTdjlrU0Rtc1A2eVU/edit?usp=sharing`

5. Participant invitation email.

   ---------- Original Message ----------
   Subject: [KDEG-list] Support needed please
   From:
   Date: Thu, June 19, 2014 7:57 am
   To:
   Cc: "Neeraj Dixit" <dixitn@tcd.ie>

   Hi all

   One of my MSc students (Neeraj Dixit) is looking for volunteers to evaluate a tool where a user generates semantic mappings between UML models. It will take less than 30 minutes of your time and you will be able to participate on-line from anywhere and anytime between 2nd July 2014 and 14th July 2013.

   All you have to do as part of the evaluation is to create mappings between a given set of UML models and then use the tool to see how many of those the tool, was able to successfully and accurately generate and finally give him some feedback on the tool.

   Please email the student directly (dixitn@tcd.ie) as soon as possible to indicate your willingness to participate

   Thanks in advance.
   Declan

6. Evaluation instruction sheet for participants.

Hi,

Below is all the information you need for the evaluation. I will be available online to help you with any questions or doubts. You can reach me on IM at

dixitn@tcd.ie
neeraj.dxt@gmail.com
and skype id : neeraj081

A help document is available for the tool at

`http://usmrt-amiorb.rhcloud.com/help`

and a demo usage video is available at

`https://drive.google.com/file/d/0B7qWHaf9smNTdjlrU0Rtc1A2eVU/edit?usp=sharing`

Please find the attached UML Diagrams showing parts of two university models. You have to create mappings in 20 minutes manually without using the tool, (you can use protege if you want) between these two and send them over to me by replying to this mail.

Once you are done with creating mappings. Please go to the below link to use the tool

`http://usmrt-amiorb.rhcloud.com/` (Please use ctr-- to resize the page to your screen size)

All you have to do is try the tool and see if it easy enough to use. Please do the following three things for sure.

(a) Use the tool to browse through generated objects and mappings.

(b) Edit any of the generated relations/mappings.

(c) Add new node and mapping.

Once you are done with the tool please fill a small Questionnaire at

`https://docs.google.com/forms/d/1zArep2-SIOJU2rPY2JjAaQcQMoXd2fOf_KMyLfPpm2g/viewform?usp=send_form`

7. Tool help document for participants.

   (a) About
   Tool to generates Semantic Mappings from UML's XMI representation.

   Features.

   i. Navigate the generated RDF graph and mappings.
   ii. Edit and update the generated RDF graph and mappings.
   iii. Download the graph in available formats.
   iv. Execute SPARQL queries against the generated graph.

   (b) Navigating Graph

   i. Once a valid XMI file is uploaded, list of objects from the generated graph will be displayed in OWL tab on the left hand side menu. Click on any object to show all its relations to other objects in graph. When an item is clicked, a graph containing Subject, predicate and objects defining the item is displayed on the right hand side window. The rectangular nodes represent the subject, objects with node name as its value and directed arrows with label are the predicate. Links in Green color represent the RDF:Type of the item and also RDF:Domain in case where the graph is of a property, with thw Diamond head representing the direaction. Links in Blue color represent the properties or RDF:Range in case of properties with Arrow head representing the direaction.
   ii. Navigating graph using relations
   Taking the cursor over any link will highlight the nodes connected by it and the relation between them. To go deeper in the relation click on the link. This will take you to screen defining that particular relation and showing that part of the complete graph.

   To get details about any of the things related to currently displayed thing node

click on the node with name. This will take you to screen showing the graph of the selected thing from the complete graph. This process can be continued till you reach the end of relations.

The path : /<name1>/<name2>/.. at the top of the graph shows the trace of the graph navigated uptill the current screen with names of the object. You can go back to any of the node in the graph by clicking on the name. Clicking on the node for which the current graph is displayed will not do anything and you will get an alert if the clicked item is a OWL construct like literal, datatype or class.Clicking on Clear from right top of menu bar will clear all the diaplayed graphs.You can also drag nodes around the window for better view.

(c) Editing Graph

The tool allows you to edit graph in two ways. First, editing or deleting the links between existing nodes. To change a relation click on the link, change the label in the pop up box and click update. To delete a relation clink on the link between nodes and click on delete button.Second, you can also add new nodes to graph by clicking on the "add Node" button and then create link between the noe node and any other node.

(d) Downloading Graph

To download the graph click on the Download button from the right top corner of the menu bar and click on the format from the list of available formats. This will download a file named mappings which you can open with any text editor.

(e) Executing SPARQL Query

SPARQL queries are executed only against the generated graph on the server.

To run a query click on the Query button at the right top corner of the menu bar. This will open a query window. write your query in the textbox, select the result format from Result Format dropdown and click on Submit.Results will be returned in TURTLE format by default if no result format is selected.

(f) Uploading UML

i. Upload from you machine.
   click on open dropdown from right top of the screen and select file. This will show a popup to select a file. You can Only upload XML files. Click on "submit" after selecting file. The tool also allows you to test a default UML. Click on show default button if you wish to see the functionality of the tool.

ii. Fetch from a URL.
   The XMI file can also be taken from a URL. Click on open dropdown from right top of the screen and select URL. Enter a valid url in the text box and click "Submit". The url must end in a valid xml file.